

A COMPARISON OF INTEGRATION METHODS FOR THE
SOLUTION OF NONLINEAR REACTOR DYNAMICS PROBLEMS
THROUGH THE USE OF FINITE ELEMENTS

Ralph Carroll Sheldrick

DUDLEY KNOX LIBRARY, --
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

NAVAL POSTGRADUATE SCHOOL

Monterey, California



THESIS

A COMPARISON OF INTEGRATION METHODS FOR THE
SOLUTION OF NONLINEAR REACTOR DYNAMICS PROBLEMS
THROUGH THE USE OF FINITE ELEMENTS

by

Ralph Carroll Sheldrick

December 1976

Thesis Advisor:

David Salinas

Approved for public release; distribution unlimited.

T176326

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Comparison of Integration Methods for the Solution of Nonlinear Reactor Dynamics Problems Through the Use of Finite Elements		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; December 1976
7. AUTHOR(s) Ralph Carroll Sheldrick		6. PERFORMING ORG. REPORT NUMBER
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		8. CONTRACT OR GRANT NUMBER(s)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		12. REPORT DATE December 1976
		13. NUMBER OF PAGES
		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Numerical methods Crank-Nicolson Gear Implicit Gear		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) A comparison of numerical methods utilized by the finite element technique for solving a nonlinear nuclear reactor dynamics problem was conducted. Using the Crank-Nicolson, DVOGER (Gear) and Implicit Gear methods, the results showed the Implicit to be the superior method investigated. This is based on the fact that all three methods yielded the same steady state solutions; but, the Implicit Gear method used		

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

significantly less CPU time and comparable storage to Crank-Nicolson. This was particularly apparent as the degrees of freedom were increased. In addition, the transient solution in all cases was better than that obtained in Crank-Nicolson and compared favorably to that of Gear's method.

The other noteworthy result was in the effect of the error criterion on solution. It was shown that for a range of error from 10^{-4} to 1.0, the steady state solution value remained the same. This results in a significant reduction in computer processing time since the time required decreases substantially as the error conditions imposed are relaxed.

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE(When Data Entered)

A Comparison of Integration Methods for the
Solution of Nonlinear Reactor Dynamics Problems
Through the Use of Finite Elements

by

Ralph Carroll Sheldrick
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1967

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1976

ABSTRACT

A comparison of numerical methods utilized by the finite element technique for solving a nonlinear nuclear reactor dynamics problem was conducted. Using the Crank-Nicolson, DVOGER (Gear) and Implicit Gear methods, the results showed the Implicit to be the superior method investigated. This is based on the fact that all three methods yielded the same steady state solutions; but, the Implicit Gear method used significantly less CPU time and comparable storage to Crank-Nicolson. This was particularly apparent as the degrees of freedom were increased. In addition, the transient solution in all cases was better than that obtained in Crank-Nicolson and compared favorably to that of Gear's method.

The other noteworthy result was in the effect of the error criterion on solution. It was shown that for a range of error from 10^{-4} to 1.0, the steady state solution value remained the same. This results in a significant reduction in computer processing time since the time required decreases substantially as the error conditions imposed are relaxed.

TABLE OF CONTENTS

I.	INTRODUCTION-----	11
A.	PURPOSE-----	11
II.	DATA GENERATORS-----	14
A.	GENERAL-----	14
B.	PROPERTY INPUTS-----	14
C.	NODAL POINT COORDINATES AND ELEMENT NODAL POINT CONNECTIVITY-----	16
D.	NODAL NEIGHBOR CONNECTIVITY-----	18
III.	CRANK-NICOLSON METHOD OF SOLUTION-----	22
A.	DESCRIPTION-----	22
B.	EFFECT OF DEGREES OF FREEDOM-----	23
C.	EFFECT OF ERROR CRITERION-----	23
D.	EFFECT OF INCREASE IN TIME STEP SIZE-----	25
IV.	DVOGER (GEAR) METHOD OF SOLUTION-----	27
A.	DESCRIPTION-----	27
B.	EFFECT OF DEGREES OF FREEDOM-----	27
C.	EFFECT OF ERROR CRITERION-----	29
V.	IMPLICIT GEAR METHOD OF SOLUTION-----	30
A.	DESCRIPTION-----	30
B.	EFFECT OF DEGREES OF FREEDOM-----	31
C.	EFFECT OF ERROR CRITERION-----	31
VI.	RESULTS-----	33
A.	COMPARISON OF TIME AND STORAGE REQUIREMENTS-----	33
B.	DEGREE OF FREEDOM EFFECT ON SOLUTION-----	34
C.	ERROR CRITERION-----	34

D. INITIAL DISTURBANCES-----	34
E. COMPARISON OF SOLUTION TRACKING-----	35
VII. TEST PROBLEMS-----	36
A. PHYSICAL MODEL-----	36
B. COMPUTER PROCESSING CONSIDERATIONS-----	36
C. PROBLEM ANALYSIS-----	36
D. PROGRAM USAGE-----	37
VIII. CONCLUSIONS AND RECOMMENDATIONS-----	39
A. CONCLUSIONS-----	39
B. RECOMMENDATIONS-----	40
TABLES-----	41
FIGURES-----	50
APPENDIX A: Computer Programming Codes-----	76
BIBLIOGRAPHY-----	143
INITIAL DISTRIBUTION LIST-----	144

LIST OF TABLES

I.	COMPARISON OF SOLUTION METHODS-----	41
II.	EFFECTS OF SOLUTION ERROR CRITERION-----	42
III.	EFFECTS OF VARYING THE INCREASE OF THE INITIAL TIME STEP IN CRANK-NICOLSON METHOD-----	43
IV.	PHYSICAL CONSTANTS-----	44
V.	LIST OF COMPUTER RUNS-----	45
VI.	DATA DECK ARRANGEMENT-----	48

LIST OF FIGURES

1.	Reactor Model with 45 Nodes-----	50
2.	Reactor Model with 72 Nodes-----	51
3.	Reactor Model with 132 Nodes-----	52
4.	Nodal Neighbor Connectivity-----	53
5.	Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Uniform Disturbance Throughout the Core-----	54
6.	Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Central Disturbance (R = 0 cm., Z = 0 cm.)-----	55
7.	Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Skewed Disturbance (R = 60 cm., Z = 0 cm.)-----	56
8.	Plot Comparing Degrees of Freedom to Computer Processing Time-----	57
9.	Effect of Time Step Change in Crank-Nicolson on Achieving Steady State-----	58
10.	Time Dependent Neutron Flux for All Methods with 45 DOF for a Central Disturbance (R = 0 cm., Z = 0 cm.)-----	59
11.	Time Dependent Neutron Flux for All Methods with 45 DOF for a Skewed Disturbance (R = 60 cm., Z = 0 cm.)-----	60
12.	Time Dependent Neutron Flux for All Methods with 45 DOF for a Uniform Disturbance Throughout Core-----	61
13.	Time Dependent Neutron Flux for All Methods with 72 DOF for a Central Disturbance (R = 0 cm., Z = 0 cm.)-----	62
14.	Time Dependent Neutron Flux for All Methods with 72 DOF for a Skewed Disturbance (R = 60 cm., Z = 0 cm.)-----	63

15.	Time Dependent Neutron Flux for All Methods with 72 DOF for a Uniform Disturbance Throughout Core-----	64
16.	Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for Central Disturbance ($R = 0$ cm., $Z = 0$ cm.)-----	65
17.	Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for Skewed Disturbance ($R = 60$ cm., $Z = 0$ cm.)-----	66
18.	Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for a Uniform Disturbance Throughout the Core-----	67
19.	Comparison of Error Criterion for Implicit Gear Method-----	68
20.	Typical Approach to Steady State Solution Value with Various Error Criterion (Implicit Gear)-----	69
21.	Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Central Disturbance ($R = 0$ cm., $Z = 0$ cm.)-----	70
22.	Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Skewed Disturbance ($R = 60$ cm., $Z = 0$ cm.)-----	71
23.	Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Uniform Disturbance Throughout the Core-----	72
24.	Time Dependent Neutron Flux for 45 and 72 DOF Using the DVOGER (Gear) Method for a Central Disturbance ($R = 0$ cm., $Z = 0$ cm.)-----	73
25.	Time Dependent Neutron Flux for 45 and 72 DOF Using the DVOGER (Gear) Method for a Skewed Disturbance ($R = 60$ cm., $Z = 0$ cm.)-----	74
26.	Time Dependent Neutron Flux for 45 and 72 DOF Using the DVOGER (Gear) Method for a Uniform Disturbance Throughout the Core-----	75

LIST OF SYMBOLS

A_{ij}	--	matrix
a	--	constant
B_{ij}	--	matrix
b	--	constant
C_{ij}	--	matrix
c	--	constant
D	--	neutron diffusion coefficient
H	--	reactor height
R	--	reactor radius
t	--	time
v	--	neutron velocity
α	--	reactivity temperature coefficient
ϵ	--	fission energy
ν	--	number of neutrons per fission
Σ_a	--	neutron absorption cross-section
Σ_f	--	neutron fission cross-section
ψ	--	neutron dynamic flux
ω	--	constant
$\bar{h}A/V$	--	modified convection heat transfer coefficient

I. INTRODUCTION

A. PURPOSE

This research project has been undertaken to compare several numerical methods of solving a nonlinear nuclear reactor dynamics problem. Three methods have been investigated in this thesis, including the Crank-Nicolson, the DVOGER (Gear) and the Implicit Gear methods of solution.

A nuclear reactor dynamics problem with temperature-dependent feedback, when it entails either a non-homogeneous or multi-region reactor, results in a nonlinear field equation in space and time. This problem does not lend itself to solution by analytical means [5, 6]. However, when the physical and neutronic properties of the problem are known, a model can be formulated using the finite element method which will yield the transient and steady state flux solutions. In particular, the partial differential equations investigated were of the form

$$a \frac{\partial \psi}{\partial t} = b \nabla^2 \psi + c \psi - \omega \psi^2 \quad (1)$$

where a , b , c , and ω are constants and $\psi(r, z, t)$ is the flux. The finite element method reduces Equation (1) to the system of ordinary differential equations

$$\sum_{j=1}^N A_{ij} \dot{\psi}_j(t) = \sum_{j=1}^N B_{ij} \psi_j(t) \quad i = 1, \dots, N \quad (2)$$

when the nonlinear term is linearized. Nguyen and Salinas [5]

and more recently Olsen [6] discuss the problem and methods of solution.

In this work, a comparison of three numerical integration schemes for Equation (2) was made. The comparisons will be made on several bases: computer storage requirements, computer processing time, rapidity with which solution was obtained and the relative accuracy of the solution.

At steady state, it is expected that all three methods will provide the same solution value. This is due to the fact that at steady state the time derivative of the flux ($\dot{\psi}$), Equation (2), is zero.

In order to explore the relative value of the various methods, the model has been discretized into finite element grids of various degrees of freedom (DOF). The effect on the solution by finer discretizations of the finite element has been investigated to determine if the various methods of solution are similarly influenced to provide a better solution for a finer mesh. To have a method of solution relatively independent of mesh size would greatly reduce computer storage and processing time requirements since a larger grid with fewer elements could be utilized.

In order to test the flexibility of the various equation solvers, an initial disturbance was introduced at different points in the model. This provided both a check of the ability of the method to accept a random disturbance as well as information on how rapidly a solution is obtained with a particular disturbance input. Two nodal points of the system

were considered, a point at the origin and a point on the core-reflector interface. This was done to determine if the tracking ability was consistent throughout the model.

Additional comparisons investigated include the effects of the convergence criterion on the solution for all methods and the effects of the size of the time step on the Crank-Nicolson method.

Modifications have been made to the programs provided in Ref. [6] which was the basis of this research project. In that work, core properties were arbitrarily assigned to the reflector elements at the interface. This occurred because the material and nuclear properties were provided at the nodal points rather than by elements. The properties were introduced in this manner as a means of computer storage reduction. Regardless of the mesh size, there were always fewer nodes than elements. In this project, all properties were provided on an element basis to eliminate this discrepancy while at the same time sacrificing the minor storage savings it represented.

II. DATA GENERATORS

A. GENERAL

The data generators formulated in this work are useable only for regular rectangular grids (Figures 1-3). Having selected the number of horizontal and vertical points for the discretization of the model, the data generators will provide the numbering of each nodal point, the horizontal and vertical position of each node and the nodal neighbors of each node.

In addition, the outer boundary nodes, at which the dynamic flux is zero, will be numbered last in order to reduce the number of equations required by the particular method of solution being used. This will substantially decrease the storage requirements. For example, in a one hundred thirty-two node discretization, only one hundred ten equations will be solved.

B. PROPERTY INPUTS

1. Purpose

A simple data generator has been provided which will produce a data deck containing the physical properties of the reactor core and reflector in the format required by the Crank-Nicolson and DVOGER (Gear) methods. These properties are neutron velocity, neutron diffusion length, neutron absorption cross-section, neutron fission cross-section and reactivity temperature coefficient.

This generator will accommodate any size mesh and will process an indefinite number of grids simultaneously.

The user must provide this routine with a data deck which contains the total number of elements in the grid and the type of each element, core or reflector. This has been simplified by the designation of all core elements by ITYPE=0 and reflector elements by ITYPE=1.

2. Programming Notes

a. The property values in the two regions must be provided as an integral part of the program.

b. The type of element, core or reflector, must be provided.

c. The routine will process an unlimited number of models. It, therefore, must be halted when all data has been utilized. This is done by specifying the final value of the number of elements in an IF-STOP statement.

3. Parameters

a. NEL - number of elements in the discretized model

b. ITYPE - type of element; ITYPE=0 is a core element and ITYPE=1 is a reflector element.

c. D - vector containing the diffusion length of the core and reflector elements.

d. SGA - vector containing the absorption cross-section of the core and reflector elements.

e. SGF - vector containing the fission cross-section of the core and reflector elements.

f. V - vector containing the neutron velocity.

g. ALPHA - vector containing the reactivity temperature coefficient.

C. NODAL POINT COORDINATES AND ELEMENT NODAL POINT CONNECTIVITY

1. Purpose

Nodal point positioning in the model is readily obtained in data deck form from this routine. Once the model to be investigated has been discretized into the finite element grid desired, the user provides the number of vertical and horizontal nodal points and their dimensional position along the axes. By the use of a nested loop, the nodal points are consecutively numbered and assigned the proper coordinate dimensions.

The element connectivity, that is, the nodal points for each element, is also resolved by the use of a nested loop and several counters. Each iteration will yield two elements with their respective nodal point boundaries. This will continue until the nodal points for each element have been computed.

The output of this routine will consist of two data decks. The first will provide the nodal point with its vertical and horizontal coordinates. The second will yield the element nodal point connectivity.

Omitted from the element connectivity data deck, but required by the thesis program, is the type of element, core or reflector. This must be added to the deck individually.

Example of output (See Figure 1):

nodal point coordinates				
nodal point	horizontal position		vertical position	
1	0.0		0.0	

element connectivity				
element	nodal point connectivity			type element*
1	1	9	10	0

*Note: must be added manually to data card.

2. Programming Notes

a. The maximum number of vertical and horizontal nodal points in the discretized model must be provided.

b. The horizontal and vertical positions of the discretization must be provided.

c. The boundary points of the model will be numbered such that these nodal points are the last in the numerical sequence.

d. The routine is written such that it will process an indefinite number of models. Therefore, it must be halted when all the data has been utilized. This will be accomplished by specifying the maximum number of vertical points in the model in an IF-STOP statement.

3. Parameters

a. NH - maximum number of horizontal nodal points in the discretized model.

b. NV - maximum number of vertical nodal points in the discretized model.

c. X - vector containing the horizontal positions of the NH points.

d. Y - vector containing the vertical positions of the NV points.

e. SYSNOD - vector providing the total number of nodal points.

f. R - vector providing the radial position of the nodal points.

g. Z - vector providing the vertical position of the nodal points.

h. ELNOD - array providing the nodal point boundaries for each element.

i. NEL - the total number of elements in the discretized model.

D. NODAL NEIGHBOR CONNECTIVITY

1. Purpose

This routine produces a data deck for the Crank-Nicolson and Implicit Gear methods containing each nodal point and the nodal points connected to it by the finite element discretization.

The regular rectangular grids are such that no more than six nodal points contribute to any one; therefore, an Nx7 array can be formulated for nodal neighbor connectivity. For example, nodal point 26 in the 112 element grid would be stored as follows (Figure 4):

26	17	18	27	35	34	25
----	----	----	----	----	----	----

and nodal point 58

58 49 50 59 57 0 0.

As written, zeros are inputted whenever a nodal point does not have six neighbors to maintain the symmetry of the matrix. This could, however, be compacted further by reading the data as a single vector with a starting point counter to indicate when the next nodal point has been reached in the sequence. This would eliminate the need for the zeros to be supplied.

This routine will provide a data deck for any rectangular grid and for an indefinite number of grids.

2. Algorithm

This algorithm provides a dense, compacted matrix which reduces the storage size required by the main program. The use of a finite element method allows a certain amount of compacting, i.e., the banded matrix. This is a function of the finite element size, that is, the greater the number of nodal points, the larger the band. The size of the band is determined by the largest difference between nodal neighbors plus one. For example, in Figure 1, node 11 has a maximum difference of $20 - 2 = 18$, while at node 16 the difference is $43 - 7 = 36$ which is the largest difference in the forty-five node system. As the number of nodal points increases, this maximum band width also increases. In Figure 2, the maximum difference is at node 16 ($67 - 7 = 60$); and, in Figure 3, the maximum difference of $124 - 10 = 114$ exists at node 22. Therefore, although the size of the system is reduced from $N \times N$ to $N \times q$, the size is not constant and may

not be small. In general, if one used banded storage, the numbering would proceed consecutively in the direction of fewest nodes. However, this would require the identification of those nodes on the boundary since they are of constant value and do not require integration.

In the particular scheme of discretization utilized in this project, no nodal point has more than six nodal point contributors. This allows the matrix to be compacted further from $N \times q$ to $N \times 7$.

3. Programming Notes

a. The total number of nodal points, the maximum number of nodal point contributors and the maximum number of horizontal and vertical nodal points must be provided.

b. The routine will satisfy any rectangular grid. However, it must be instructed when a sufficient number of nodal points have been generated. This is accomplished by using the total number of nodal points desired in an IF-GO TO statement.

c. This routine requires a positive means of stopping. This is accomplished by the use of the maximum number of vertical nodal points in the last data set in an IF-STOP statement.

d. The output of this routine will place the central nodal point first with the contributors following.

4. Parameters

- a. NV - number of vertical nodal points in the model.
- b. NH - number of horizontal nodal points in the model.

c. NVH - total number of nodal points in the model.

$NVH = NV \times NH$.

d. LCON - maximum number of contributing nodal points.

(LCON = 7).

e. MNOD - an array, NVH x LCON, providing the central and contributing nodal points.

III. CRANK-NICOLSON METHOD OF SOLUTION

A. DESCRIPTION

The Crank-Nicolson formulation is a numerical method originally presented by J. Crank and P. Nicolson in 1947. Crank-Nicolson, being an implicit method, does not require the inverse of the matrix to be calculated. Therefore, advantage is taken of the sparse matrix inherently provided by the finite element method. If the inverse of the sparse matrix is formed, it will be full.

The Crank-Nicolson method will solve the set of linear differential equations represented by

$$\sum_{j=1}^N A_{ij} \dot{\psi}_j(t) = \sum_{j=1}^N B_{ij} \psi_j(t) \quad i=1, \dots, N. \quad (3)$$

Crank-Nicolson, after some algebra, yields

$$A \left[\frac{\psi_t - \psi_{t-\Delta t}}{\Delta t} \right] = C \left[\frac{\psi_t + \psi_{t-\Delta t}}{2} \right]. \quad (4)$$

The implicit system (4) may be solved by an iterative process; in this case, the Gauss-Seidel method is used to solve the set of simultaneous algebraic equations (4) formulated. Of all the stable numerical methods in the case of single step implicit, Crank-Nicolson has the smallest truncation error [2]. The Gauss-Seidel method of iteration is continuously using the newest solution values allowing convergence to the solution to be the most rapid.

B. EFFECT OF DEGREES OF FREEDOM

As might be expected, as the grid size became finer, the computer core requirements increase. In addition, more time is required for the problem to reach a steady state solution. At the same time, as shown in Figures 5-7, the solution values converged, as the mesh size became finer, to a more accurate solution.

As shown in Table I and Figure 8, storage and processing time, that is, CPU time to reach a steady state solution, increased markedly between a forty-five node grid and a one hundred thirty-two node grid. This yielded a fifty-nine percent increase in storage and better than a five hundred percent increase in processing time. At the same time there is an improvement in the solution of only four and one-half percent.

Comparison to a seventy-two node grid yielded far more satisfying results. At seventy-two nodes, there is an eighty percent increase in CPU time and a sixteen percent increase in storage requirements while obtaining a two and one-half percent improvement in the solution.

Similar results were obtained in problem time to solution. For one hundred thirty-two nodes, the time more than doubles; while for seventy-two nodes, the time was increased by twenty percent.

C. EFFECT OF ERROR CRITERION

The error criterion used throughout this thesis informs the integration routine when it has achieved sufficient

agreement between iterates (i.e., when it can stop and go to the next iteration step).

In Crank-Nicolson, the error criterion simply consisted of the difference in successive Gauss-Seidel iterates divided by the current iterative value.

The error criterion was varied from 10^{-1} to 10^{-4} to observe the effects on the solution and computer requirements. For Crank-Nicolson the steady state solution value remained the same regardless of the error criterion. This demonstrated that the extra time required to satisfy a tighter error criterion at each iteration was not necessary to reach a satisfactory steady state solution. As was expected, the computer processing time did increase significantly as the error criterion was made smaller. An unusual result occurred when the error criterion was set to 10^{-4} . This particular value resulted in a processing time less than that required for 10^{-1} . It would appear that this might have been the result of two contributing factors. With the closeness of the error, each time an iteration was performed, it was using a better solution, and each new time step also had a better solution value. Although there were many more time steps required for this error criterion, the steady state value was rapidly approached because fewer iterations were required at each new time. These results are shown in Table II. These results for an error criterion between 10^{-1} and 10^{-4} do not imply that this would be the result obtained for all problems of this type.

D. EFFECT OF INCREASE IN TIME STEP SIZE

In this method, the time step can be increased or not depending on the solution. If a solution is not obtained with a particular time step, that same time step is reduced; and the routine attempts to attain solution with the smaller time step. This continues until either a satisfactory solution value is obtained or the built in default value is reached which terminates the routine. When a solution is attained, the routine compares the number of iterations required to reach that value to that of the previous time step and to a specified number of iterations. If the current number of iterations is less than either of those numbers, the program increases the time step by an input constant value.

The change made to the initial time step as the steady state was approached was the critical value if the solution was to converge to a steady state. It appeared that Crank-Nicolson was particularly sensitive to the amount the time step was increased as the solution was approached (Figure 9). The method was unable to recover if, when nearing the knee of the curve, the size of the increase was such that the steady state value was exceeded. Once the solution was passed, the results indicated a diverging oscillation about the steady state value.

Several values of the initial time step were utilized in all grids to attempt to locate the steady state. The results of these trials are provided in Table III. It is noteworthy

that if the increase was greater than 1.2 times the initial step size, the steady state would never be achieved. Whether this would be the case for all grids is not obvious. This is particularly evident in view of the results obtained by Olsen [6], who utilized an increase of 1.5 times the initial step for a thirty-eight node grid. It would, therefore, seem that a trial and error approach would be necessary until an increase in step size resulted in a steady state solution.

IV. DVOGER (GEAR) METHOD OF SOLUTION

A. DESCRIPTION

The DVOGER (Gear) routine is an IMSL library routine which integrates a system of explicit first order differential equations. Within this library routine is the capability of solving both stiff and nonstiff systems by selection of an indicator. In solving the nonstiff system, the Adams predictor-corrector is utilized. For the stiff system, Gear's predictor-corrector is used. Gear's method computes the Jacobian for the system of ordinary differential equations in order to optimize the time step for each integration.

B. EFFECT OF DEGREES OF FREEDOM

Gear's method placed a much greater demand on the computer in terms of processing time and storage than the other methods considered. This was primarily due to three causes: 1) the calculation of the Jacobian, 2) the transformation of Equation (3) to explicit form, and 3) the initial absolute value of the solution. The first two items require a substantial increase in both CPU time and computer storage. Several variances were implemented in this program to determine if better use could be made of the method to make it competitive with Crank-Nicolson or the Implicit Gear methods.

When the system was considered nonstiff, the storage requirements decreased since the Jacobian was not calculated; but, the processing time increased by one order of magnitude

when compared to the stiff system treatment due to the small time steps taken to solution. The routine calls for an initial absolute solution value of one to be supplied. This, however, adversely affects the progress of the problem since this value of one is initially compared to values of 10^{16} . This limits the time step taken by the routine initially and continues to affect the progress until the steady state solution is neared. When an initial value of 10^{14} was utilized, the processing time was decreased by forty percent. This was due to the fact that larger time steps were allowed from the outset, since the previous solution value (in this case the initial value) was comparable to the value obtained in the first calculation.

At best, when utilizing Gear's method, the processing times were two orders of magnitude greater than that required for the Implicit Gear method and twenty times the Crank-Nicolson processing times for the 132 DOF system, as shown in Table I. Core requirements were also increased significantly, particularly at one hundred thirty-two degrees of freedom. For this grid, DVOGER (Gear) was approximately three times Crank-Nicolson and the Implicit Gear core requirements.

Gear's method provided the same steady state solution values as the other two methods, and the transient curves were very similar to those obtained in the Implicit Gear method (Figures 10-15).

C. EFFECT OF ERROR CRITERION

In Gear's method, the time step size is adjusted so that the single step error estimate divided by the previous maximum solution value is less than the error criterion in the Euclidean norm. The single step error estimate is a multiple of the difference between the predicted and corrected values of the variable.

The error criterion, Table II, was varied for this method as it was for the others. The results were the same; as the criterion was tightened, CPU time increased. In this case, however, the time became excessive very rapidly, more than an hour for 10^{-3} and almost four hours for 10^{-4} for the model with 45 DOF. Similarly, the solution values were the same regardless of the criterion utilized.

V. IMPLICIT GEAR METHOD OF SOLUTION

A. DESCRIPTION

This numerical method is particularly useful for the solution of large, sparse systems of implicit, stiff differential equations [4]. In contrast to Gear's method, the Implicit Gear method eliminates the need of an exact $N \times N$ Jacobian, but rather requires only an exact $N \times 7$ Jacobian. This is due to the fact that Equations (3) are handled directly, thereby retaining the sparseness of the matrix.

Gear's predictor-corrector method and the compact matrix makes use of storage because the implicit Equations (3) are handled directly. This results in very efficient use of the computer both in terms of storage and processing time requirements.

The user is required to provide a subroutine that evaluates the system of equations being investigated, as well as, for efficiency, a subroutine to evaluate the Jacobian of the system of equations.

This program is a modification by Franke [4] to the routine DFASUB [7]. One of the major changes to the routine is in the treatment of the error. In the Implicit Gear method, instead of using the Euclidean norm, the root mean square norm is utilized. This is no more than the Euclidean norm divided by the square root of the number of components. In addition, the maximum value of the component is updated before the norm of the relative error is computed.

B. EFFECT OF DEGREES OF FREEDOM

The steady state values obtained in this method were the same as those obtained in Crank-Nicolson and DVOGER (Gear) as shown in Figures 11-18. This result does not imply that these methods will always give identical steady state solutions. Certainly they do give different transient solutions. The major advantages of this method are 1) the computer processing time was reduced as the grid size became finer and 2) the storage requirements are slightly greater than that required in the Crank-Nicolson method due to the computation of the Jacobian and the storage of up to seven past solutions which control the size of the time step. Implicit Gear requires about $N \times 25$ more storage locations than Crank-Nicolson. This amounts to about only thirteen thousand bytes for the one hundred thirty-two degrees of freedom system (single precision). The magnitudes of the increase in processing time dropped dramatically in this method. For the same initial disturbance, the time increased eighty percent for the one hundred thirty-two node grid and twenty-four percent for the seventy-two degrees of freedom. In addition, Implicit Gear gave a more accurate transient solution. The results of this method are shown in Table I and comparison to the other methods will be made in Chapter VI.

C. EFFECT OF ERROR CRITERION

In Implicit Gear, the error criterion is defined as in DVOGER (Gear) except that the root mean square of the Euclidean

norm and the updated maximum solution value are used in the computation.

The error criterion was varied substantially to determine what effect it had on computer storage and time requirements, as well as its effect on the accuracy of solution. As shown in Table II and Figures 19-20, a wide range of convergence, 1.0 to 10^{-4} , was investigated with interesting results. Expectedly, the processing time did increase as a tighter error criterion was required. However, the criterion had little effect on the track through the transient solution for values of less than 10^{-1} . When using an error of one-half and one, the transient solution varied substantially. The same steady state solution was obtained, although at a later point in problem time as shown in Figure 19. For the stiffer error criterion requirements, the processing time necessary is doubled by utilizing a criterion of 10^{-4} instead of a value greater than 10^{-1} . This, certainly, did not appear to warrant the closer tolerance level.

VI. RESULTS

A. COMPARISON OF TIME AND STORAGE REQUIREMENTS

With all three methods yielding the same steady state solution (Figures 10-18), the comparison of methods became one of time and storage requirements placed on the computer rather than one based on which provided the best solution. In the transient stage, Gear tracked slightly better than Implicit Gear, but both Gear and Implicit Gear tracked better than Crank-Nicolson (See Figures 10-18). For the three systems of equations utilized, the Implicit Gear method performed significantly better than either Crank-Nicolson or DVOGER (Gear) in CPU time and was comparable to Crank-Nicolson and superior to Gear in storage requirements (Table I). This becomes more significant as the number of degrees of freedom increase. The Implicit Gear is less sensitive to the increase in size of the system of equations as shown in Figure 8.

Comparing the forty-five and the one hundred thirty-two degrees of freedom, the core requirements increased slightly and the time doubled for the Implicit Gear method. In contrast, for Crank-Nicolson, the time increased by six times and the core by 100K bytes; and, for DVOGER (Gear), there was a ten times and 500K byte increase in time and storage respectively.

B. DEGREE OF FREEDOM EFFECT ON SOLUTION

Varying the degrees of freedom resulted in a better solution value as shown in Figures 5-7 and 21-26. There was a four and one-half percent better solution obtained for the one hundred thirty-two degrees of freedom compared to the forty-five degrees of freedom. This better solution is very costly both in terms of computer core requirements and CPU time (Figure 8).

C. ERROR CRITERION

Error criterion was varied for all methods to determine the effect on solution and computer. As might be expected, the computer processing time increased for all methods as more rigid tolerance was imposed (Table II). However, although the criterion was made very close, there was no appreciable effect on the transient solution until the error criterion was greater than 10^{-1} and no effect on the steady state solution. This is significant in that, as stated above, the processing time increases greatly as the criterion is tightened. This shows that for the particular problem considered here, some close error criteria yield no advantage, only the disadvantage of requiring more time in the computer.

D. INITIAL DISTURBANCES

The input of various initial disturbances (central, skewed at the core-reflector interface and uniform throughout the core) yielded the same steady state solution value. The transient solution varied substantially due to the input

position of the disturbance. In most cases, due to the magnitude and scope of the initial disturbance, the uniform disturbance required the least processing time to reach the steady state value.

E. COMPARISON OF SOLUTION TRACKING

Two nodal points were investigated graphically for each disturbance, integration method and DOF, as shown in Figures 5-7, 10-18 and 21-26. At these test points, the methods performed similarly in all cases in both the transient and steady state solutions.

VII. TEST PROBLEMS

A. PHYSICAL MODEL

A cylindrical reactor with the dimensions and properties given in Table IV was used as a model. A radial slice of this model was discretized by various finite element grids as shown in Figures 1-3.

B. COMPUTER PROCESSING CONSIDERATIONS

The programs presented in this thesis, Appendix A, were written in the FORTRAN IV language and all computer runs, Table V, processed on the IBM 360/67 computer using the FORTRAN 'G' compiler. It is expected that the FORTRAN 'H' compiler would have supplied results similar to those obtained with the FORTRAN 'G' compiler. This was not done because the FORTRAN 'H' compiler requires 350K bytes as a minimum core requirement. The majority of the runs performed in this thesis required significantly less than 300K storage. Single precision (six to seven significant digits) was used throughout this research. As has been shown [6], double precision solutions are at variance by less than 0.01 percent with single precision solutions.

C. PROBLEM ANALYSIS

The techniques used by Olsen [6], modified as described, and the Implicit Gear method [4] were utilized to solve the problem delineated in Ref. [5]. The problem was approached

using finite element discretizations with forty-five, seventy-two and one hundred thirty-two degrees of freedom and providing an initial disturbance. Three initial disturbances were provided as follows: central at the core center ($R = 0$ cm., $Z = 0$ cm.); uniform throughout the core; and, a skewed disturbance at the core-reflector interface ($R = 60$ cm., $Z = 0$ cm.).

D. PROGRAM USAGE

In order to properly utilize the routines presented in this thesis, several points must be considered.

In the Crank-Nicolson routine, there was difficulty in obtaining a steady state solution. The ultimate cause was the size with which the previous time step increases. This requires a trial and error approach for the particular grid size. For this project, an increase of ten or twenty percent yields satisfactory results; but, anything larger results in a divergent oscillation about the steady state value.

In all three methods, the dimensioning should be set by the particular problem being solved, i.e., determined by the degrees of freedom. If this is done, most efficient use will be made of the computer, and the user will experience better turn around time on the equipment.

In the use of the Implicit Gear method, the data cards for nodal neighbor connectivity must have the contributing nodes listed consecutively, with the zeros, used for nodes not having six neighbors, being last on the cards. For example, in the one hundred thirty-two degrees of freedom system

(Figure 3), the nodal neighbor connectivity for node 122 would be written as

122	123	11	0	0	0	0.
-----	-----	----	---	---	---	----

The data deck arrangement for all three methods is given in Table VI.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

It has been shown in this research project that of the three methods investigated in the solution of the test problem that the Implicit Gear performed in a superior manner in CPU time requirements and was comparable to the storage requirements of Crank-Nicolson. In addition, Implicit Gear was the least sensitive to the change in the degrees of freedom.

The method used had no effect on the steady state solution value obtained as all three yielded the same results for the same DOF. In the transient solution, Implicit Gear conformed very closely to DVOGER (Gear). As shown in Figures 10-15, as the DOF increased, the transient solutions moved in the direction of the transient solution of Gear's method.

As the DOF in the discretized finite element was increased, a better solution was obtained. This, however, was counteracted by the fact that for the small increase in accuracy obtained by the finer mesh, the time and storage need of the computer increased significantly.

A very important result of this project was the effect of varying the error requirements of the problem. This yielded essentially no change in the accuracy of the transient solution for a range of 10^{-1} to 10^{-4} and no variance in the steady state solution for a range from 1.0 to 10^{-4} . In using these methods of integration, the error criterion selected would

be a function of the solution desired. If the user is only concerned with the steady state solution, the tolerance could be relaxed to 1.0. However, if the transient solution is needed, an error of 10^{-1} appears to be the least rigid value which will still provide a satisfactory track to steady state.

B. RECOMMENDATIONS

It would be of value to investigate further the Implicit Gear method through the use of other problems solving a non-linear system of ordinary differential equations. In addition, larger degrees of freedom should be attempted to evaluate any limitations that may exist in this method of integration.

TABLE I
COMPARISON OF SOLUTION METHODS

METHOD	CORE (K)	PROCESSING TIME* (min.)		
		Central	Skewed	Uniform
45 Nodes				
CRANKO	158	1.67	1.73	3.57
DVOGER (GEAR)	148	17.8	19.5	12.5
IMPLICIT GEAR	124	1.05	1.00	1.20
72 Nodes				
CRANKO	184	3.07	5.49	2.38
DVOGER (GEAR)	244	89.5	114.1	59.5
IMPLICIT GEAR	124	1.3	1.30	1.25
132 Nodes				
CRANKO	252	9.1	9.1	9.1
DVOGER (GEAR)	610	>400	>400	>400
IMPLICIT GEAR	124	1.9	2.0	2.0

* Processing time is that required to reach a steady state solution.

TABLE II
EFFECTS OF SOLUTION ERROR CRITERION

45 Nodes

SOLUTION ERROR CRITERION	PROCESSING TIME (min.)	TIME AT SOLUTION (sec.)
CRANK-NICOLSON		
0.1	1.8	3.69×10^{-5}
0.01	5.5	3.70×10^{-5}
0.001	14.2	3.57×10^{-5}
0.0001	0.83	3.87×10^{-5}
IMPLICIT GEAR		
1.0	0.99	7.56×10^{-4}
0.5	0.92	1.48×10^{-2}
0.1	1.05	7.06×10^{-5}
0.01	1.20	7.64×10^{-5}
0.001	1.45	6.36×10^{-5}
0.0001	1.58	6.44×10^{-5}
DVOGER (GEAR)		
0.1	18.6	5.64×10^{-5}
0.01	29.5	5.79×10^{-5}
0.001	78.8	5.55×10^{-5}
0.0001	>300	—

TABLE III
EFFECTS OF VARYING THE INCREASE OF THE INITIAL TIME STEP
IN CRANK-NICOLSON METHOD

INCREASE	CORE	PROCESSING TIME (min.)	SOLUTION TIME (sec.)
<u>45 NODES</u>			
1.1	158K	4.99	3.18×10^{-5}
1.2	158K	1.65	2.69×10^{-5}
1.5	184K	7.42	OSCILLATING
<u>72 NODES</u>			
1.1	252K	3.5	4.38×10^{-5}
1.2	184K	3.05	3.23×10^{-5}
1.3	252K	3.5	OSCILLATING
1.4	252K	9.1	OSCILLATING
1.5	184K	6.1	OSCILLATING
<u>132 NODES</u>			
1.1	252K	9.1	5.58×10^{-5}
1.2	252K	8.4	8.03×10^{-5}
1.5	184K	2.6	OSCILLATING

TABLE IV
PHYSICAL CONSTANTS

SYMBOL	COMPUTER SYMBOL	DEFINITION	VALUE
C-N, D / IG			
R	R	Total radius	90 cm.
R _c	R	Core radius	60 cm.
H _c	Z	Core height	160 cm.
H	Z	Total height	220 cm.
v	V/VELOCT	Neutron velocity	4.8×10^7 cm/sec
D _c	D/DSUBF	Neutron diffusion coefficient (core)	0.913 cm.
D _r	D/DSUBC	Neutron diffusion coefficient (reflector)	1.20 cm.
Σ_{ac}	SGA/SGMAAF	Neutron absorption cross-section (core)	0.01401 cm^{-1}
Σ_{ar}	SGA/SGMAAC	Neutron absorption cross-section (reflector)	0.008 cm^{-1}
v	ZNU	Number of neutrons per fission	2.54
Σ_{fc}	SGF/SGMAF	Neutron fission cross-section (core)	0.008 cm^{-1}
Σ_{fr}	SGF/ -	Neutron fission cross-section	0.0 cm^{-1}
ϵ	FISFAC	Fission energy	7.652×10^{-12} cal/fis
\bar{h}_A/V	HBAR	Modified convection heat transfer coefficient	$0.0632 \text{ cal/cm}^3\text{-sec-}^\circ\text{C}$
α	ALPHA	Reactivity temperature coefficient	$1 \times 10^{-5} / ^\circ\text{C}$

TABLE V
LIST OF COMPUTER RUNS

RUN	METHOD	NODES	DISTURBANCE	COVERGENCE	TIME STEP CHANGE	
1	CRANK-NICOLSON	45	CENTRAL	0.1	1.1	
2					1.2	
3					1.5	
4				0.01	1.2	
5				0.001		
6				0.0001		
7			UNIFORM	0.1		
8			SKEWED			
9		72	CENTRAL	0.1	1.0	
10					1.1	
11					1.2	
12					1.3	
13					1.4	
14					1.5	
15			UNIFORM		1.2	
16			SKEWED			
17		132	CENTRAL	0.1	1.1	
18					1.2	
19					1.5	
20					UNIFORM	1.2
21					SKEWED	
22	DVOGER MTH=0				45	CENTRAL
23	DVOGER MTH=1 YMAX=1	0.1				

TABLE V (Continued)

RUN	METHOD	NODES	DISTURBANCE	CONVERGENCE	TIME STEP CHANGE
24				0.01	
25				0.001	
26	DVOGER MTH=1 YMAX=1	45	CENTRAL	0.0001	
27			UNIFORM	0.1	
28			SKEWED		
29		72	CENTRAL		
30			UNIFORM		
31			SKEWED		
32		132	CENTRAL		
33			UNIFORM		
34			SKEWED		
35	IMPLICIT GEAR	45	CENTRAL	0.1	
36				0.01	
37				0.001	
38				0.0001	
39			UNIFORM	0.1	
40			SKEWED		
41		72	CENTRAL	0.1	
42			UNIFORM		
43			SKEWED		
44		132	CENTRAL	0.1	
45			UNIFORM		
46			SKEWED		
47		45	CENTRAL	0.5	

TABLE V (Continued)

RUN	METHOD	NODES	DISTURBANCE	CONVERGENCE	TIME STEP CHANGE
48				1.0	
49	DVOGER MTH=1 YMAX=10 ¹⁴	45	CENTRAL	0.1	

TABLE VI
DATA DECK ARRANGEMENT

CRANK-NICOLSON

Title

NUMEL,NUPBP,NUMSNP,NFULEL

List of outer boundary points

MTH,MAXDER,NCOUNT

ZNU,FISFAC,HBAR,EPSVAL,ERRVAL,AFUEL

TO,H,TF,HMIN,HMAX

V - neutron velocity

D - diffusion length

SGA - absorption cross-section

SGF - fission cross-section

ALPHA - reactivity temperature coefficient

PSIIV - initial disturbance flux

System nodal points with axial and radial position

Element nodal connectivity

Nodal neighbor connectivity

DVOGER

Title

NUMEL,NUPBP,NUMSNP,NFULEL

List of outer boundary points

MTH,MAXDER,NCOUNT

ZNU,FISFAC,HBAR,EPSVAL,ERRVAL,AFUEL

TO,H,TF,HMIN,HMAX

V

TABLE VI (Continued)

D

SGA

SGF

ALPHA

PSIIV

System nodal points with radial and axial position

Element nodal connectivity

IMPLICIT GEAR

NCORE

List of core elements

with uniform initial disturbance

Title

NUMEL,NBP,NUMSNP,NFULEL,NELROW,MXEVAL,NCOUNT,NCMPK

NDE,NL

List of outer boundary points

VELOCT,DSUBF,DSUBC,SGMAAF,SGMAAC,SGMAF,FISFAC,HBAR

ZNU,ALPHA,RMSEPS,AFUEL,TSTART,TEND,PINITV,SCALE

HMIN,HMAX

MTH,MAXDER,NPROB,NTYPE,JPLOT,NPLOT,IRUN

Nodal neighbor connectivity

System nodal points with radial and axial position

Element nodal connectivity

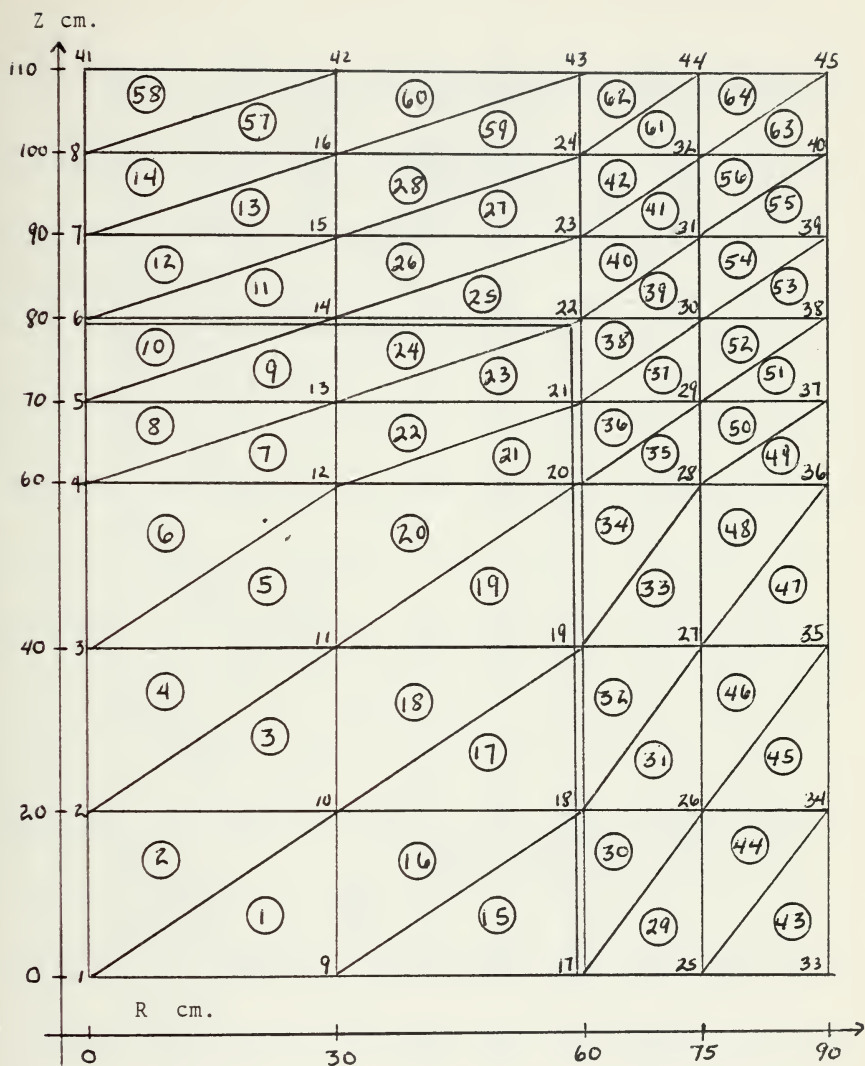


Figure 1. Reactor Model with 45 Nodes

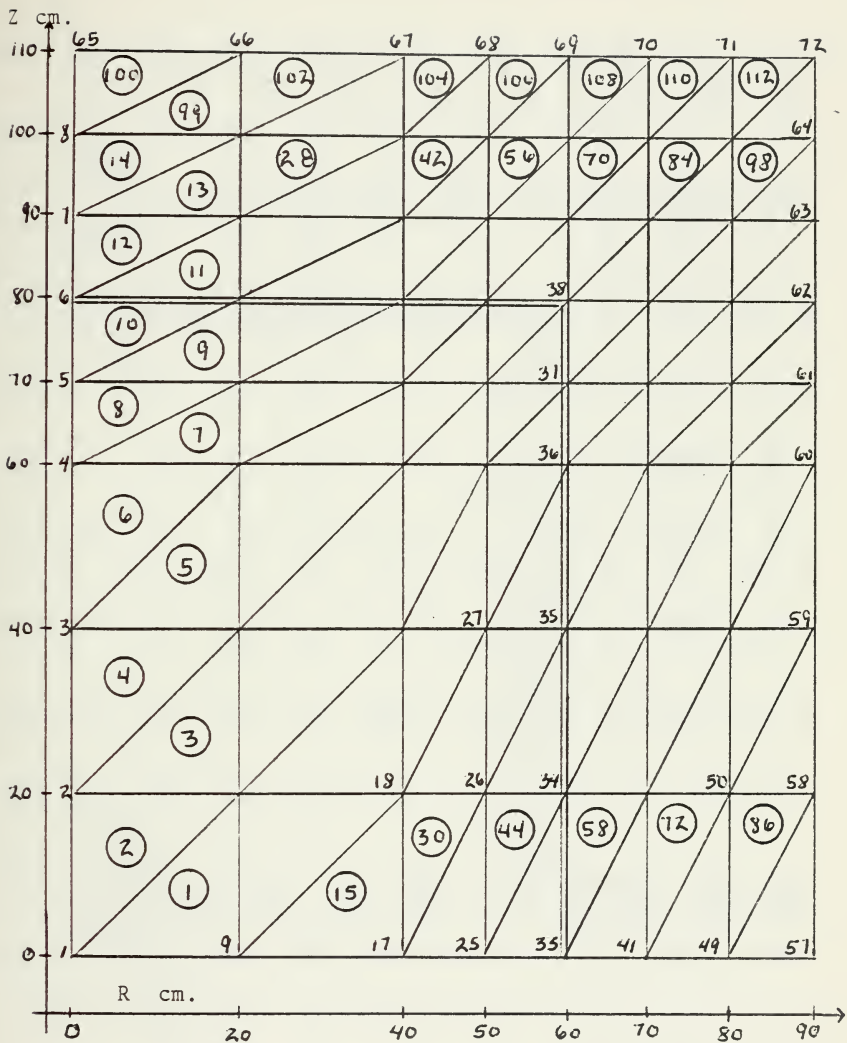
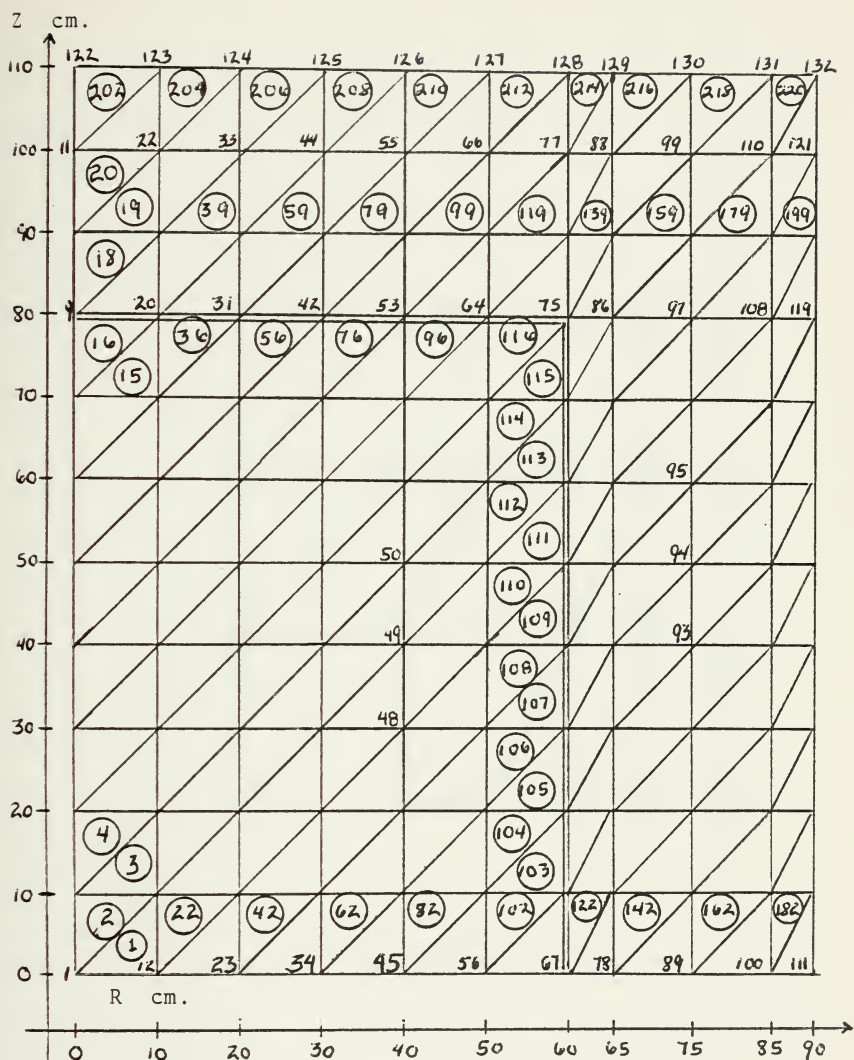


Figure 2. Reactor Model with 72 Nodes



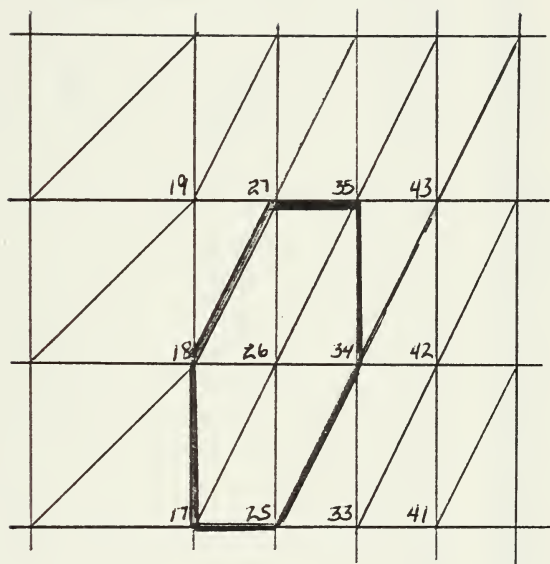


Figure 4. Nodal Neighbor Connectivity

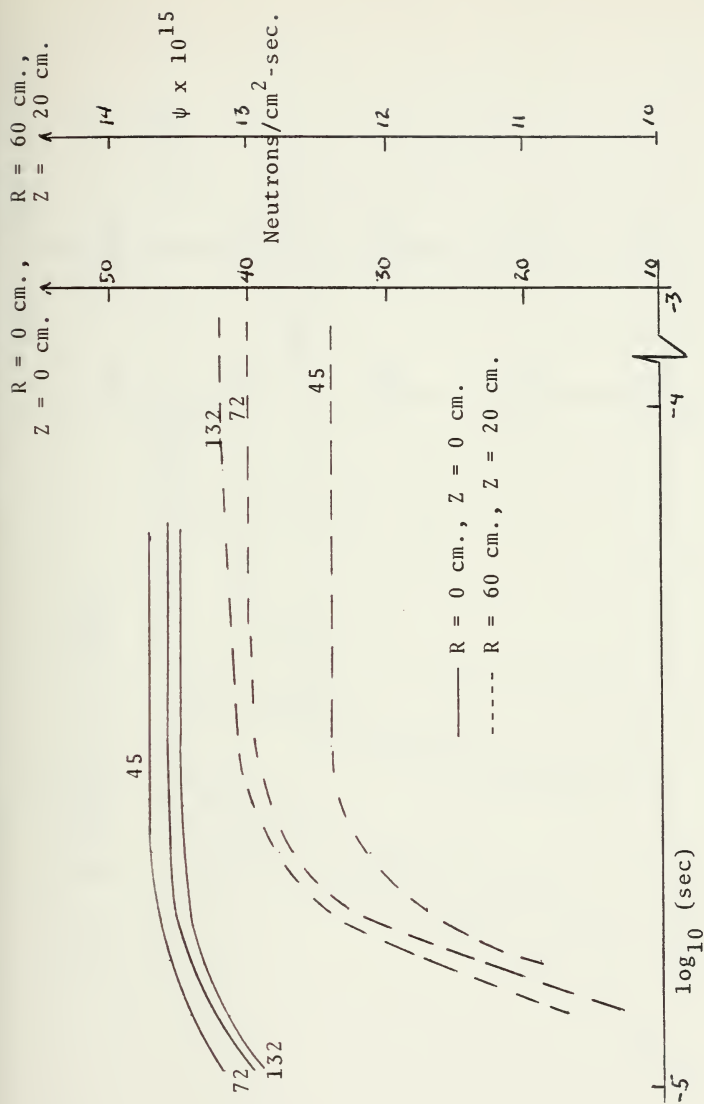


Figure 5. Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Uniform Disturbance Throughout the Core.

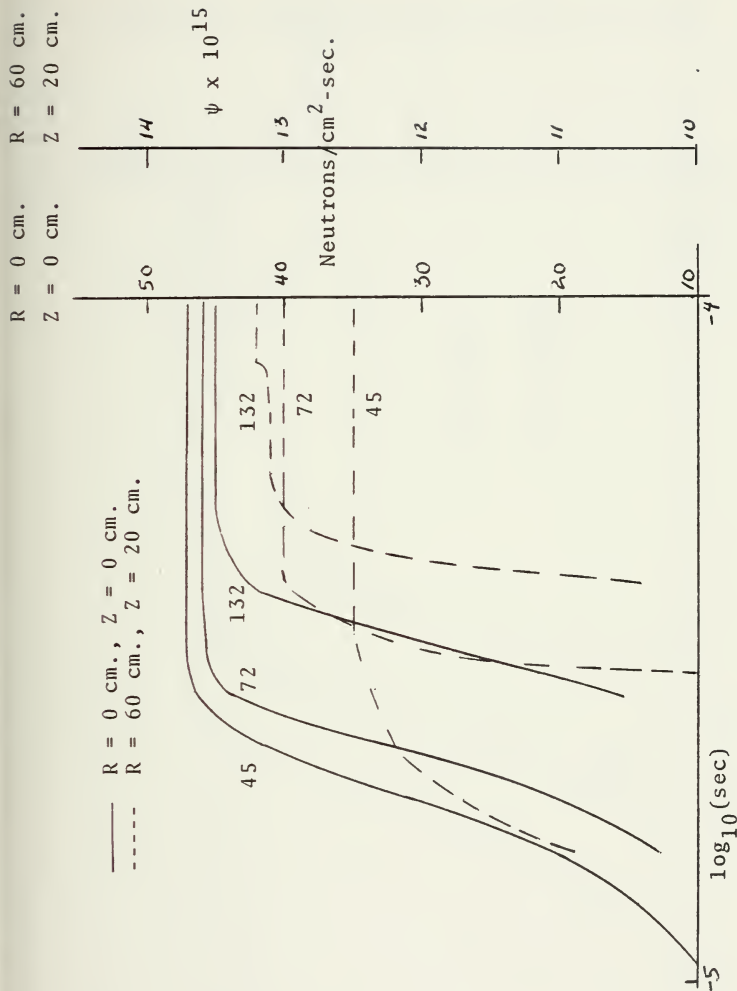


Figure 6. Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Central Disturbance ($R = 0 \text{ cm., } Z = 0 \text{ cm.}$)

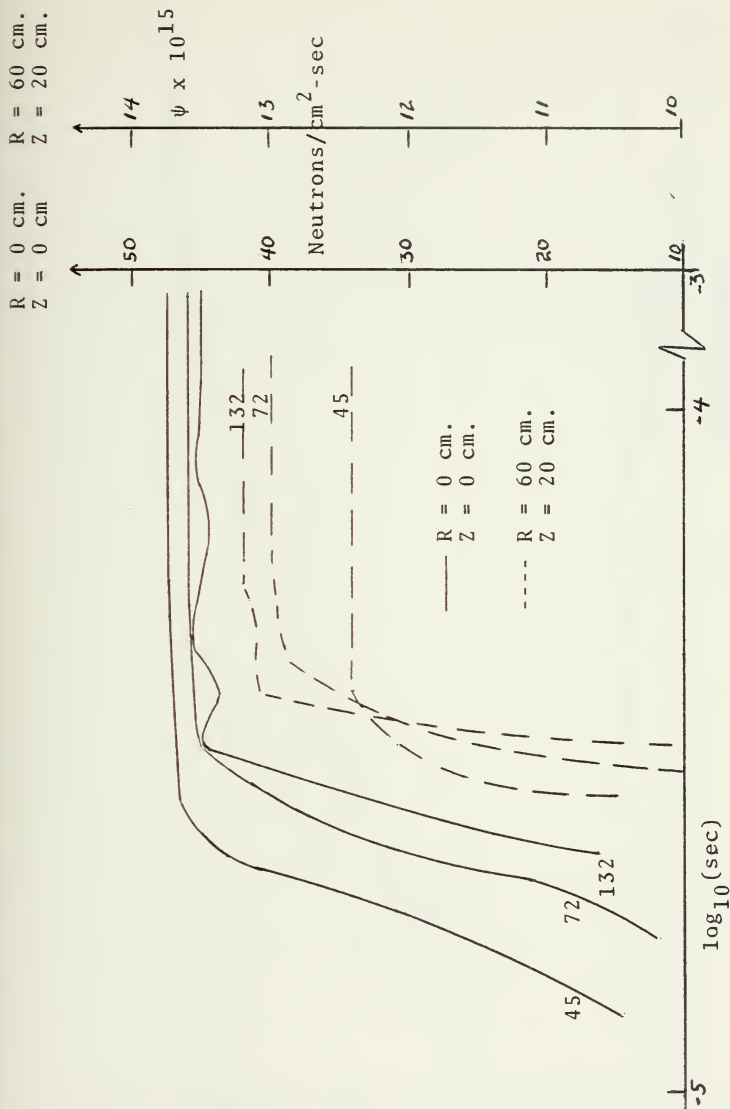


Figure 7. Time Dependent Neutron Flux for All DOF Using Crank-Nicolson Method for a Skewed Disturbance ($R = 60 \text{ cm.}$, $Z = 0 \text{ cm.}$)

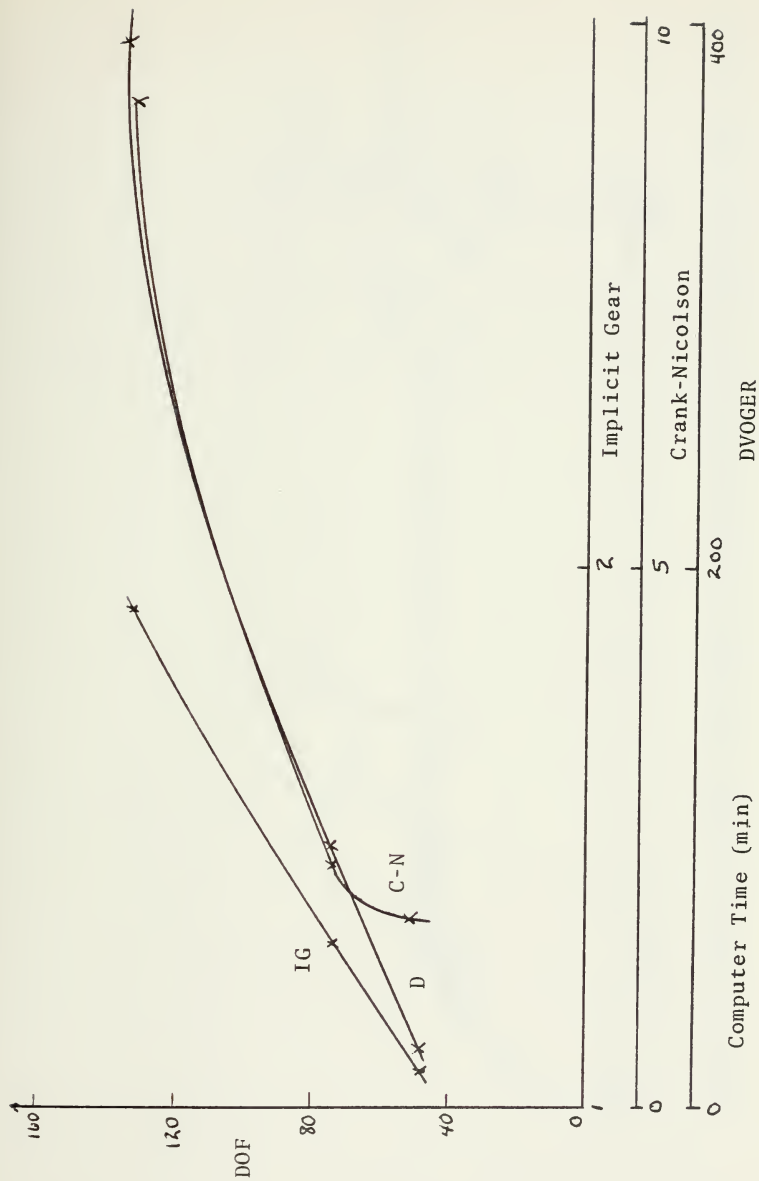


Figure 8. Plot Comparing Degrees of Freedom to Computer Processing Time

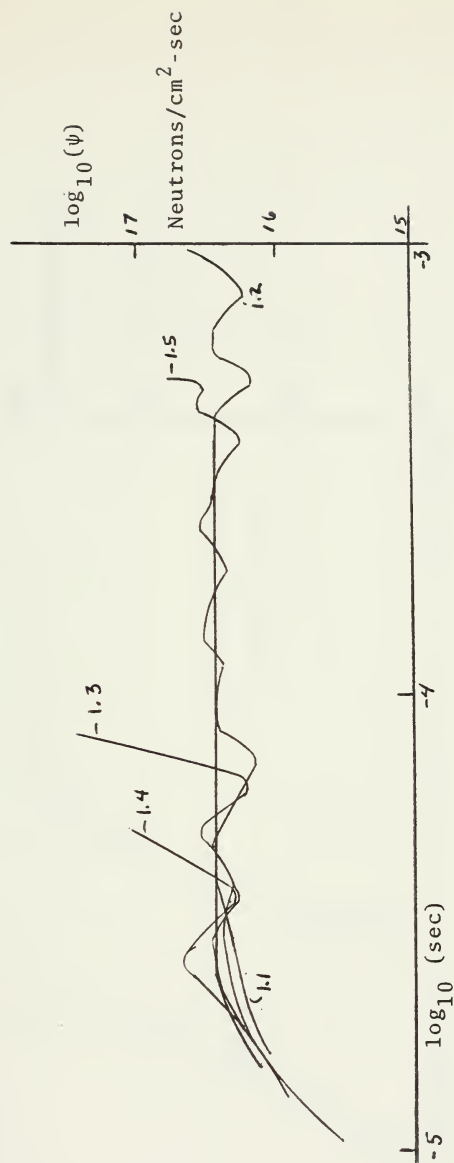


Figure 9. Effect of Time Step Change in Crank-Nicolson on Achieving Steady State.

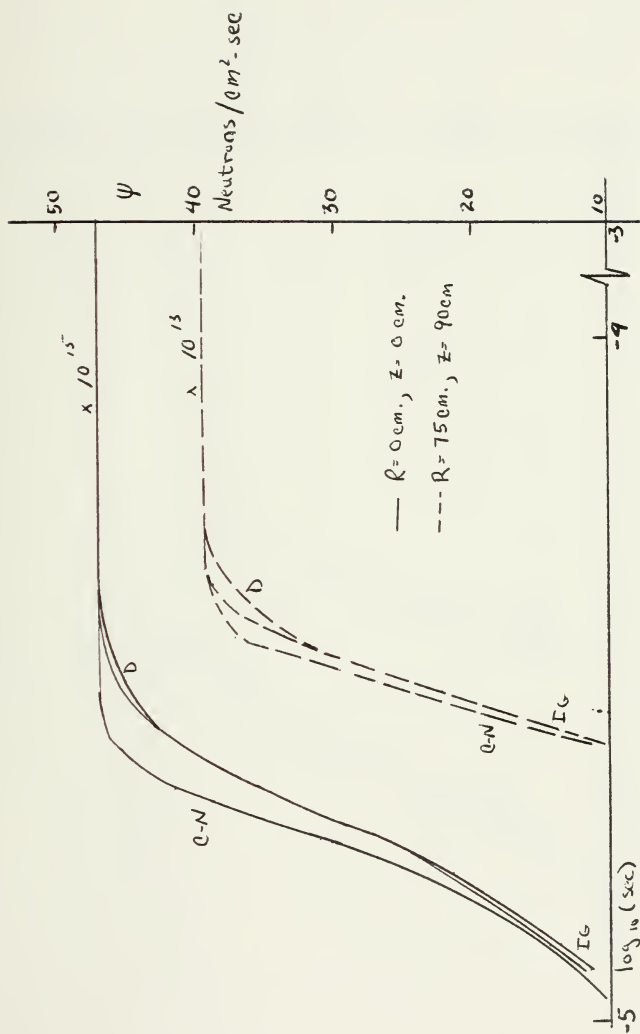


Figure 10. Time Dependent Neutron Flux for All Methods with 45 DOF for a Central Disturbance ($R = 0$ cm., $Z = 0$ cm.).

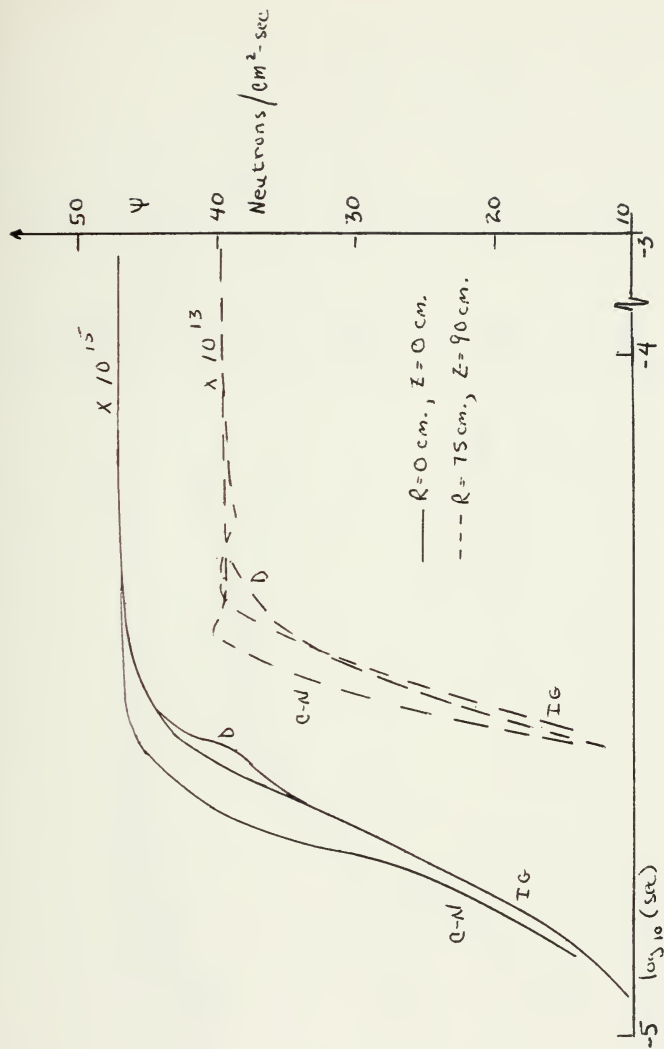


Figure 11. Time Dependent Neutron Flux for All Methods with 45 DOF for a Skewed Disturbance ($R = 60$ cm., $Z = 0$ cm.).

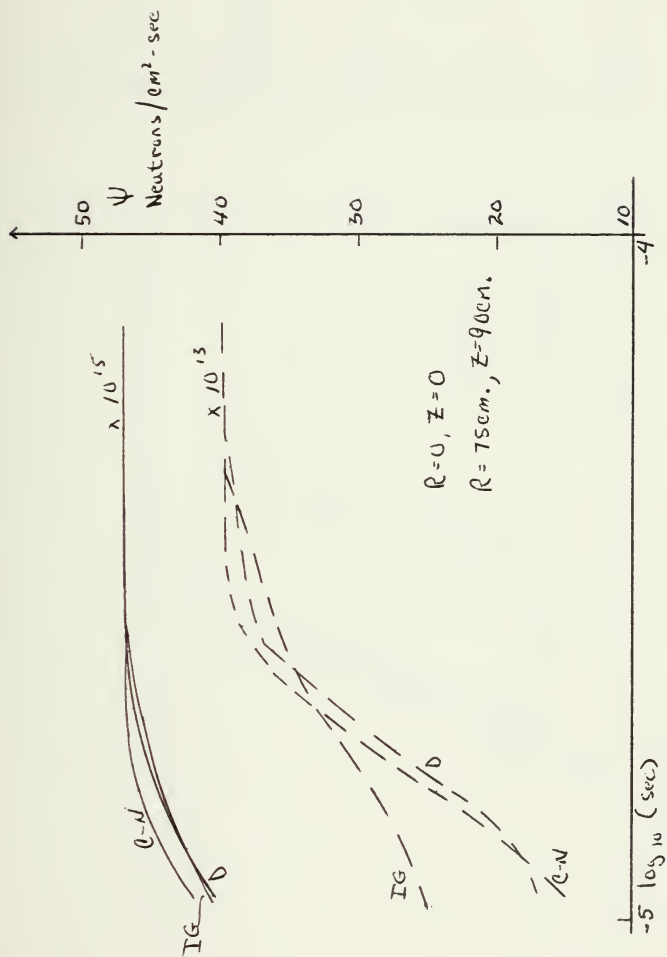


Figure 12. Time Dependent Neutron Flux for All Methods with 45 DOF for a Uniform Disturbance Throughout Core.

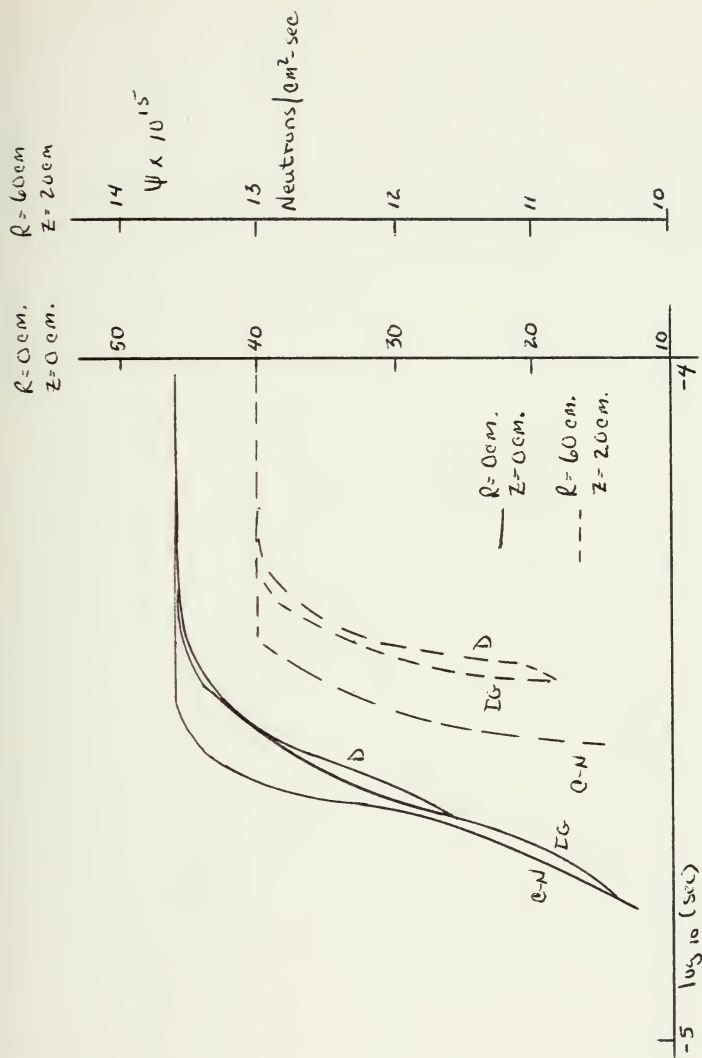


Figure 13. Time Dependent Neutron Flux for All Methods with 72 DOF for a Central Disturbance ($R = 0 \text{ cm.}$, $Z = 0 \text{ cm.}$).

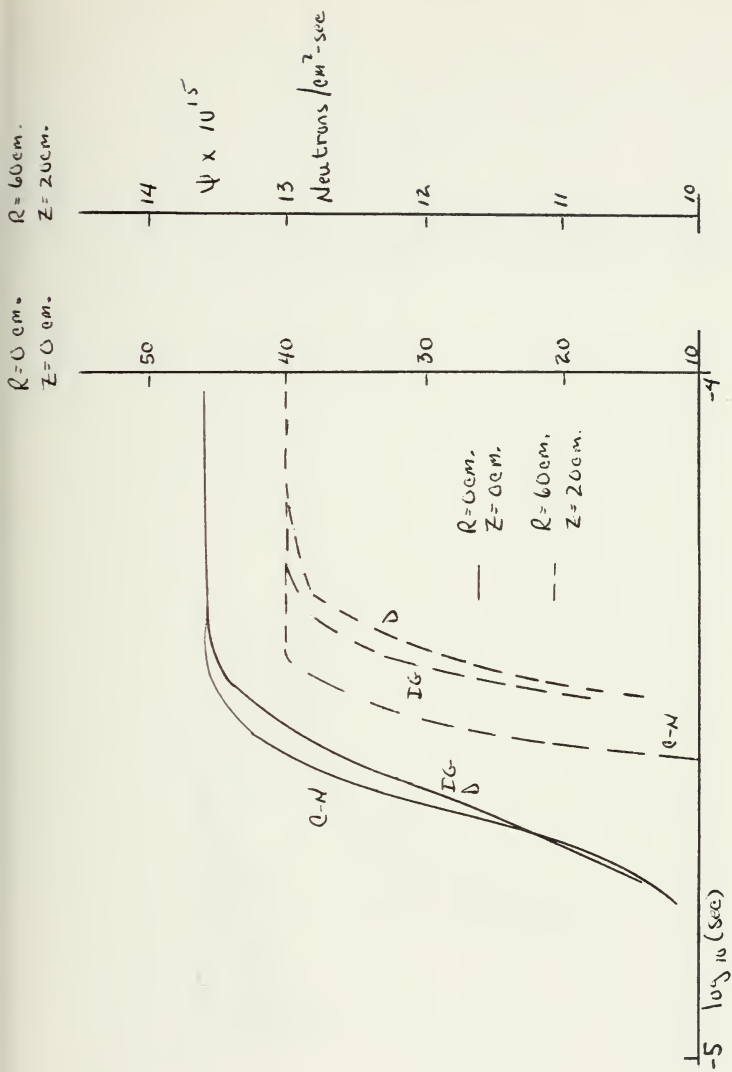


Figure 14. Time Dependent Neutron Flux for All Methods with 72 DOF for a Skewed Disturbance ($R = 60 \text{ cm.}$, $Z = 0 \text{ cm.}$).

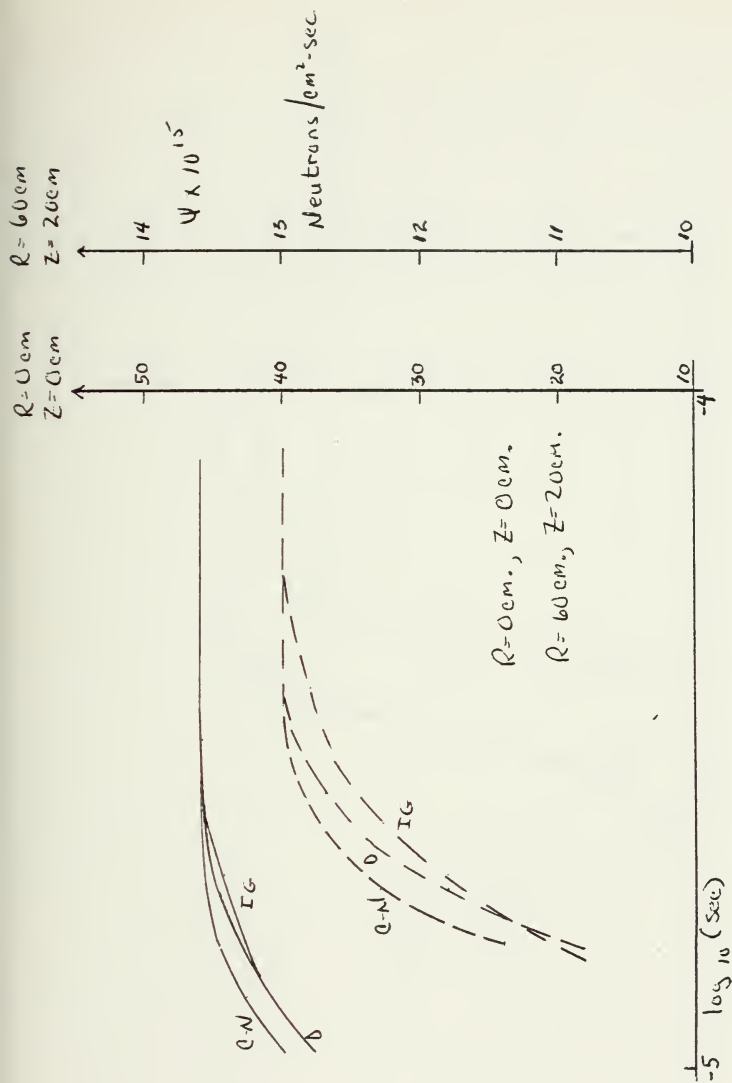


Figure 15. Time Dependent Neutron Flux for All Methods with 72 DOF for a Uniform Disturbance Throughout Core.

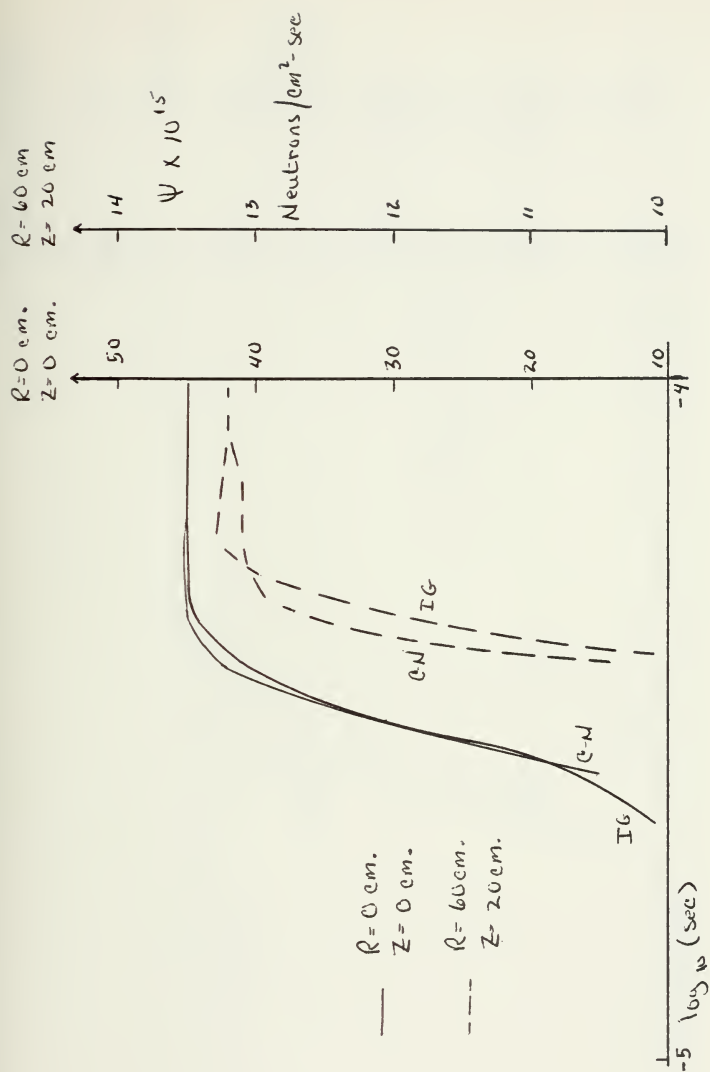


Figure 16. Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for Central Disturbance ($R = 0 \text{ cm.}$, $Z = 0 \text{ cm.}$)

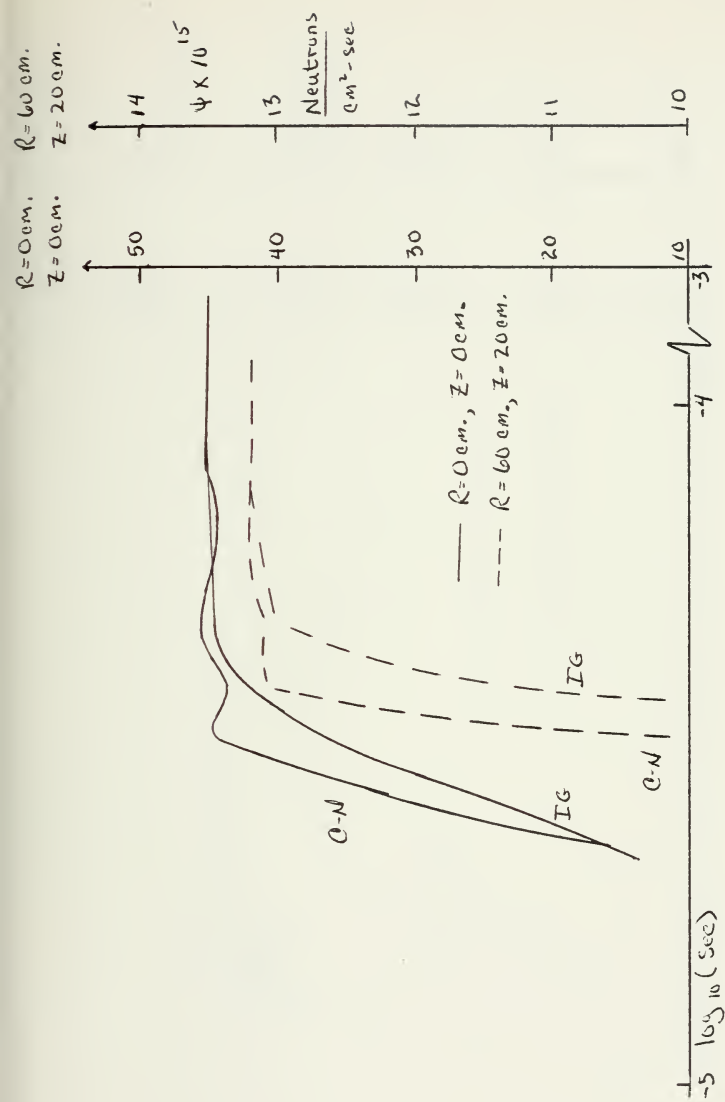


Figure 17. Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for Skewed Disturbance ($R = 60 \text{ cm.}$, $Z = 0 \text{ cm.}$)

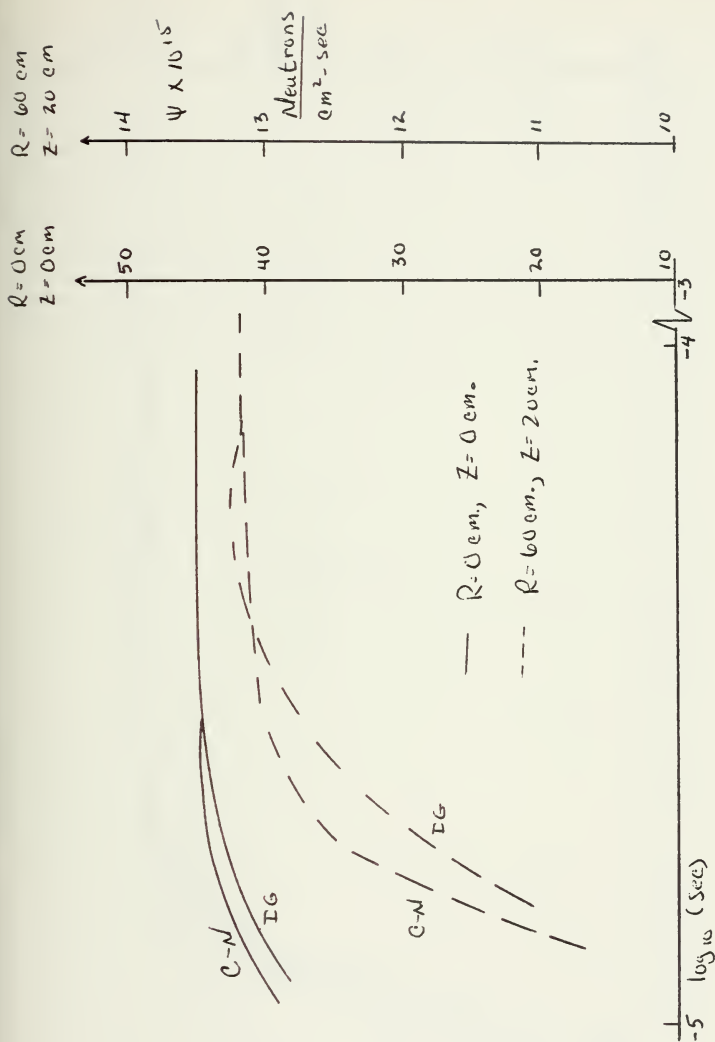


Figure 18. Time Dependent Neutron Flux for Crank-Nicolson and Implicit Gear Methods with 132 DOF for a Uniform Disturbance Throughout the Core.

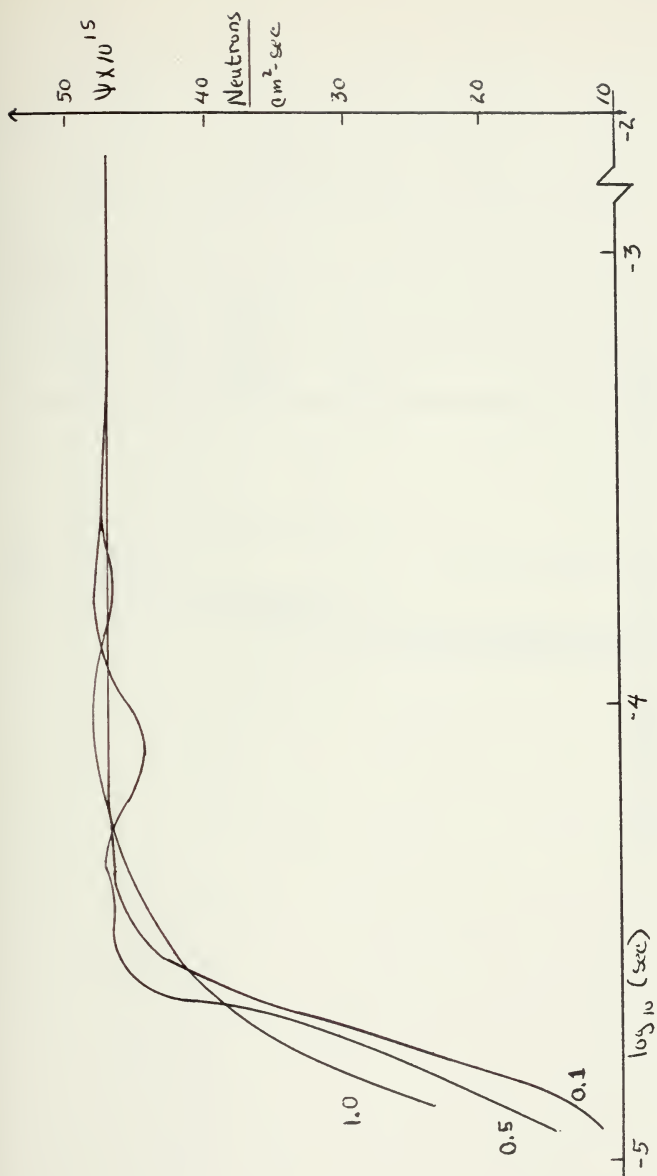


Figure 19. Comparison of Error Criterion for Implicit Gear Method.

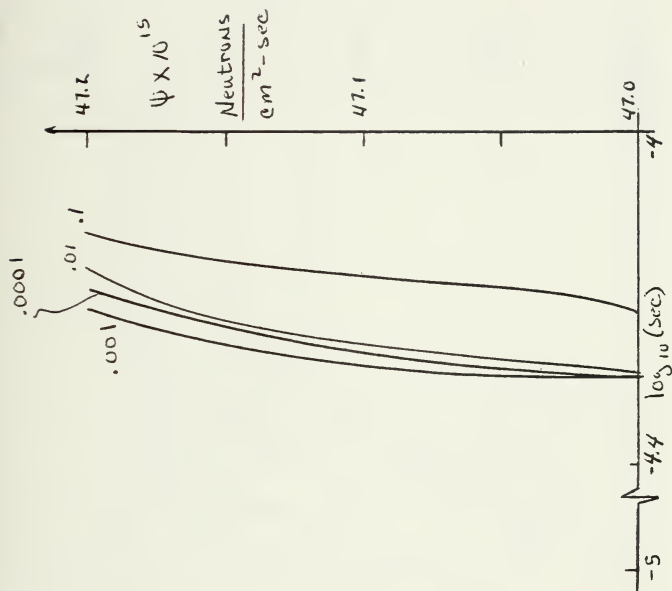


Figure 20. Typical Approach to Steady State Solution Value with Various Error Criterion (Implicit Gear).

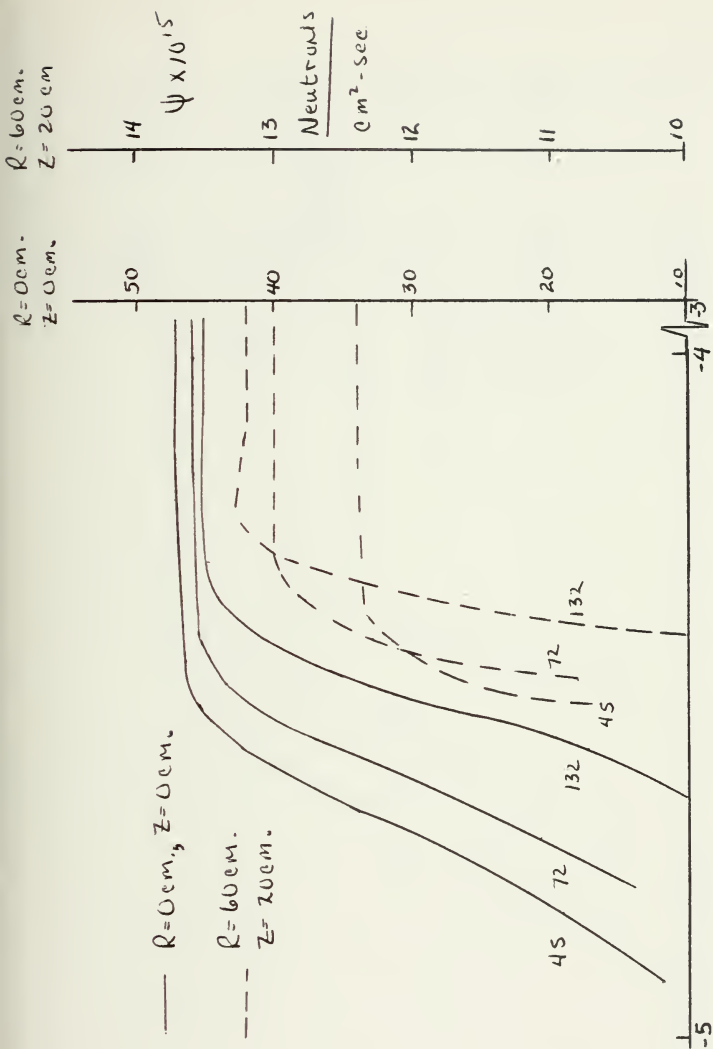


Figure 21. Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Central Disturbance ($R = 0 \text{ cm.}, Z = 0 \text{ cm.}$)

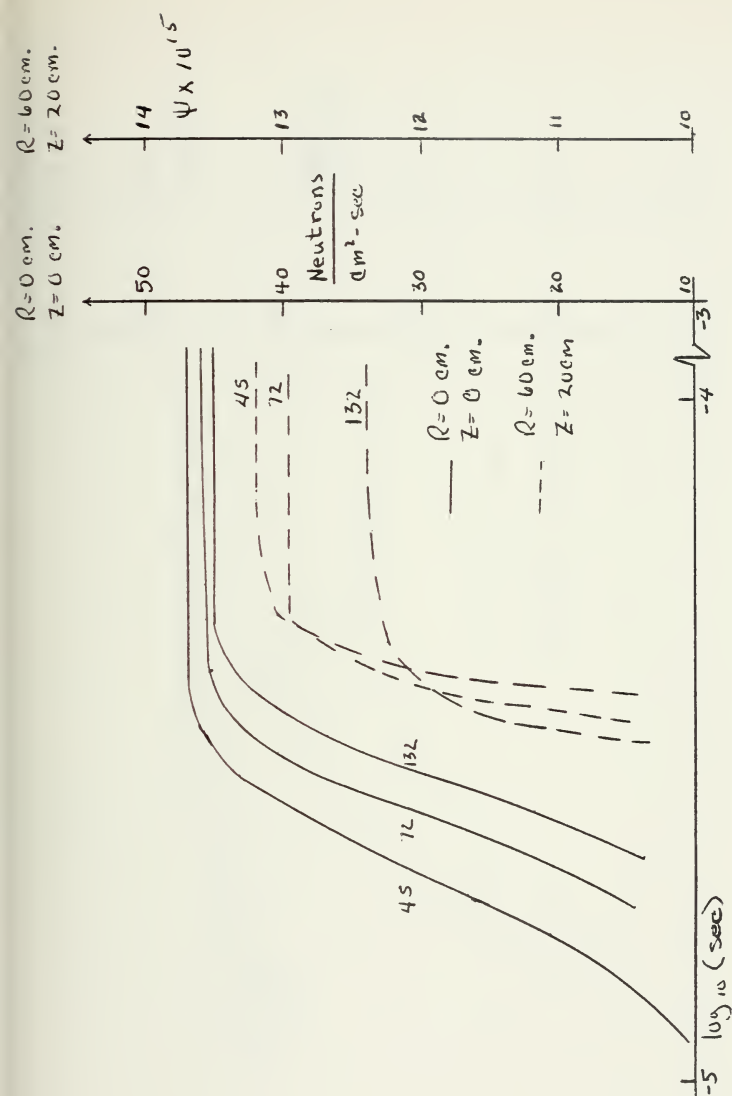


Figure 22. Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Skewed Disturbance ($R = 60 \text{ cm.}$, $Z = 0 \text{ cm.}$).

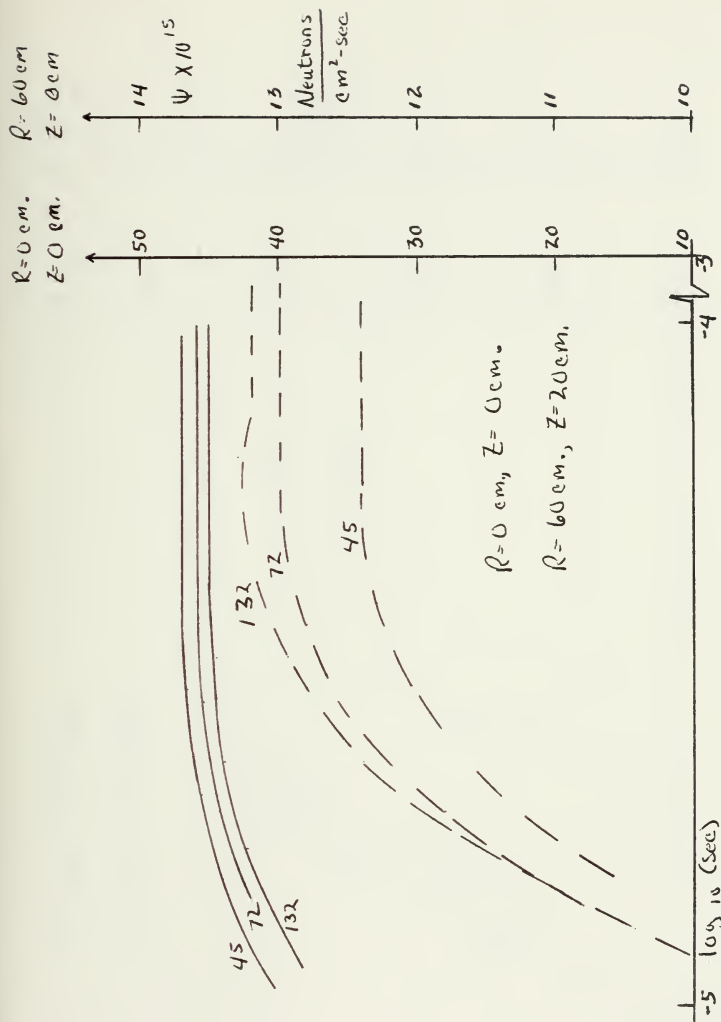


Figure 23. Time Dependent Neutron Flux for All DOF Using Implicit Gear Method for a Uniform Disturbance Throughout the Core.

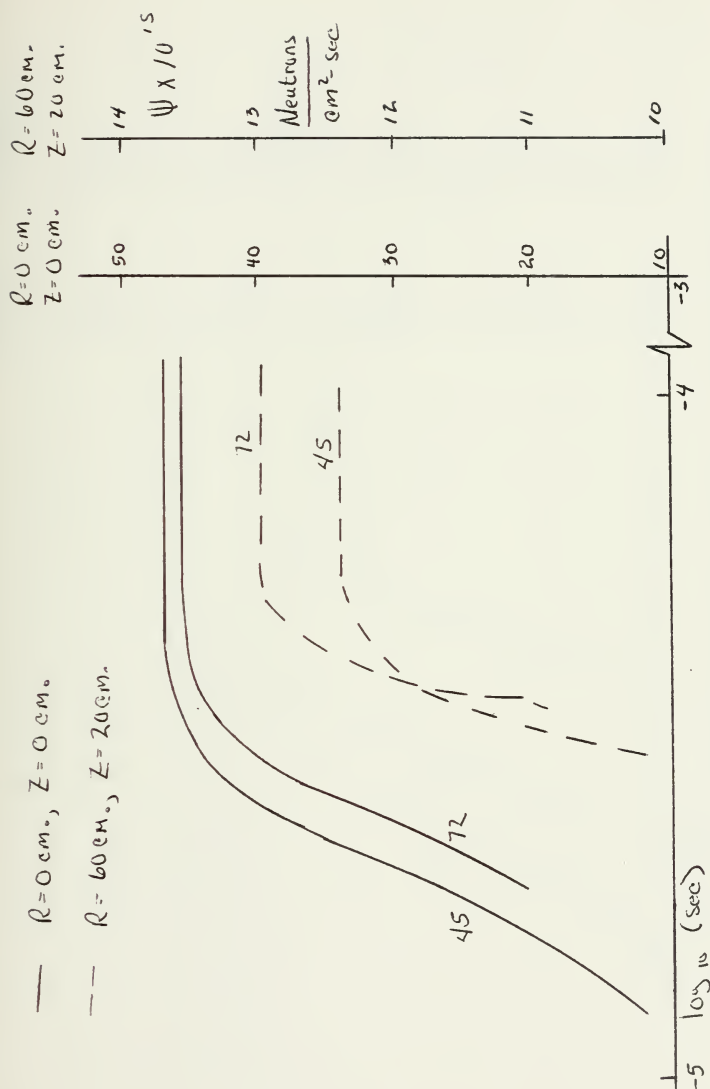


Figure 24. Time Dependent Neutron Flux for 45 and 72 DOF Using the DVOGER (Gear) Method for a Central Disturbance ($R = 0 \text{ cm.}, Z = 0 \text{ cm.}$).

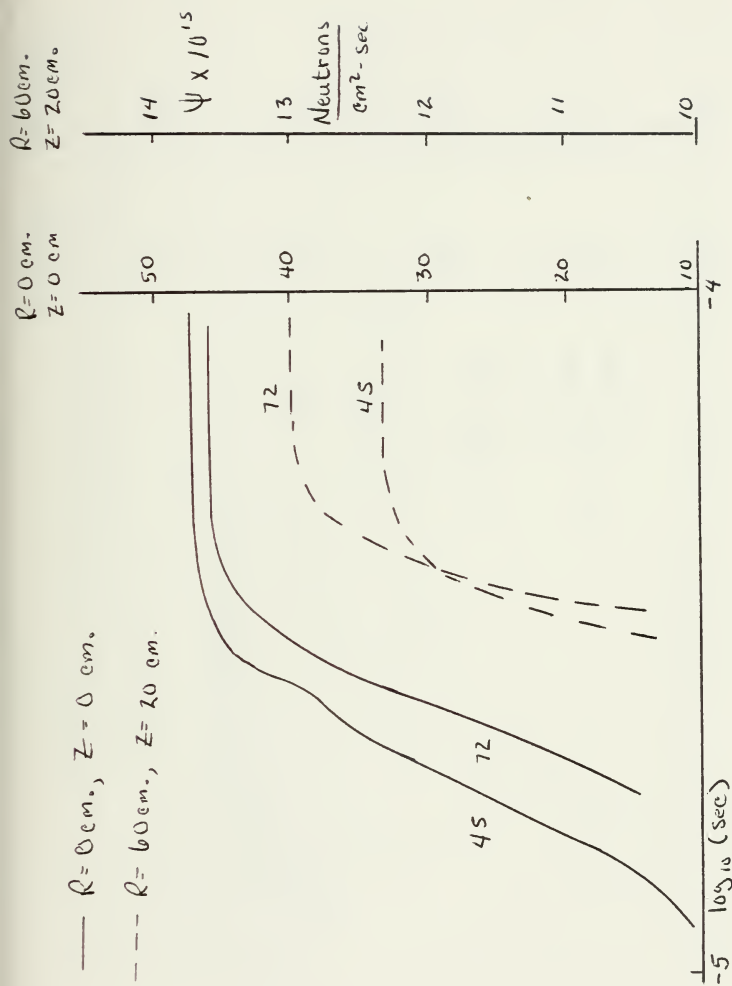


Figure 25. Time Dependent Neutron Flux for 45 and 72 DOF Using the
 DVOGER (Gear) Method for a Skewed Disturbance
 ($R = 60 \text{ cm.}$, $Z = 0 \text{ cm.}$).

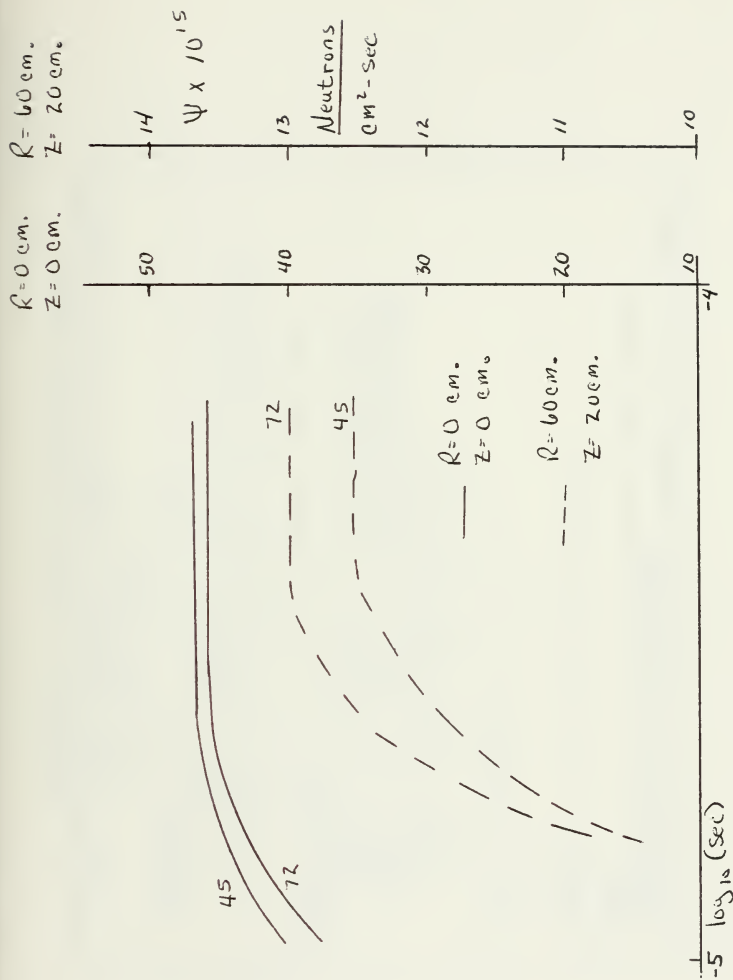


Figure 26. Time Dependent Neutron Flux for 45 and 72 DOF Using the DVOGER (Gear) Method for a Uniform Disturbance Throughout the Core.

APPENDIX A: Computer Programming Codes

```
*****  
**          FINITE ELEMENT SOLUTION OF NONLINEAR REACTOR DYNAMICS  
**          IN TWO DIMENSIONAL SPACE  
**          LINEARIZED VERSION  
*****  
  
SOLUTION METHOD -- DVJGER (IMSL)  
OPTIONS  
    1- MTH=0 INDICATES A PREDICTOR CORRECTOR  
        (ADAMS) METHCD  
    2- MTH=1 INDICATES A VARIABLE ORDER METHOD  
        SUITABLE FOR SYSTEMS OF STIFF DIFFERENTIAL  
        EQUATIONS. PROGRAM USES THE JACOBIAN.  
*****  
  
INPJT PARAMETERS AND DIMENSIONING REQUIREMENTS.  
SINGLE ITEMS  
NUMSNP IS THE NUMBER OF SYSTEM NODAL POINTS  
NUMNIP IS THE NUMBER OF TRIANGULAR ELEMENTS  
VUPBP IS THE NUMBER OF OUTER BOUNDARY POINTS  
NFUEL IS THE NUMBER OF ELEMENTS IN THE CORE  
EPSVAL IS THE CONVERGENCE CRITERIA  
ITYPE=0 DENOTES CORE ELEMENTS  
ITYPE=1 DENOTES REFLECTOR ELEMENTS  
*****  
  
DIMENSIONING REQUIREMENTS  
NUMSNP BY NUMSNP  
PART,BIGA,BIGAB,BIGC,BIGCC,WPI  
*****  
NUMSNP  
SYSVOD R,Z,YMAX,ERROR,PSIIV,DPSI,PV  
*****  
NUMEL  
ELEMNT, ITYPE,V,V0,SGA,SGF,ALPHA,D,ZLMBDA,VLMBDA,  
ZOMEGA,R1,R2,R3,Z1,Z2,Z3,AREA  
ELNOD(NUMEL,3),IJPOP(IJNPBP),CM(NUMEL,3,3,3),PW(NWK)  
*****
```



```

C      INTEGER*4  SYSNOD,ELEMNT,ELNOD
C      DIMENSION  TITLE(20),ALFA(3,3)

C      DIMENSION  PART(132,132),BIGA(132,132),BIGAB(132,132),BIGC(132,132)
C      1,BIGCC(132,132),SYSNOD(132),ELEMNT(220),ELNOD(220,3),IUPBP(22,
C      2,IITYE(220),V(220),VD(220),D(220),SGA(220),SGFL(220),A,PHA(220),
C      3,ZLMBDA(220),VLMBDA(220),ZOMEGA(220),RHO(220),R(132),Z(132),
C      4,R1(220),Z1(220),R2(220),Z2(220),R3(220),Z3(220),A(220,3),
C      5,B(220,3),YMAX(132),ERROR(132),AREAT(220),CM(220,3,3),DP,SI(132)
C      6,PSIIV(132),POPSI(132),PW(20000),PSI(8,132),PV(132),S(132),
C      7,WP(132,132)

C      READ(5,998) TITLE
C      READ(5,10) NUMEL,NUPBP,NUMSNP,NFULEL
C      998  FORMAT(20A4)
C      10  FORMAT(8I10)
C      *   FOR MTH NOT EQUAL TO ZERO  NWK = NJM*SNP * (NUMSNP + 17)
C      NWK=NUMSNP*17
C      CALL ERRSET(207,256,0,1,1,209)

C      DO 05 I=1,NUMSNP
C      ERROR(I)=0.
C      PDPSI(I)=0.
C      YMAX(I)=1.
C      05  CONTINUE

C      NUF=NUMSNP/5
C      XNP=NUMSNP
C      XNUF=XNP/5
C      IF ((XNUF-FLOAT(NUF)) .LT. 1.)  NUF=NUF+1

C      GENERAL PROBLEM DATA
C      READ(5,12) (IJPBP(I),I=1,NUPBP)
C      READ(5,10) MTH,MAXDER,NCOUNT
C      12  FORMAT(16I5)

C      WRITE(6,11) NJMEL,NUPBP,NUMSNP,NFULEL
C      11  FORMAT(7/2X,'# OF ELEMENTS=',15//2X,'# OF UPPER BDRY NODAL POINTS=
C      2I5)',15//2X,'# OF SYSTEM NODAL POINTS=',15//2X,'# OF FUEL ELEMENTS=',
C      READ(5,15) ZNJ,EISFAC,HBAR,EP,SVAL,ERRVAL,AFUEL
C      READ(5,19) TO,H,TF,HMIN,HMAX

```



```

READ(5,15) (V(I),I=1,NUMEL)
READ(5,15) (D(I),I=1,NUMEL)
READ(5,15) (SGA(I),I=1,NUMEL)
READ(5,15) (SGF(I),I=1,NUMEL)
READ(5,1000) (ALPHA(I),I=1,NUMEL)
READ(5,18) (PSIIV(I),I=1,NUMSNP)
1000 FORMAT(5F15.8)
15 FORMAT(8G10.5)

```

```

C
C
C
C
CALCULATE PHYSICAL CONSTANTS FROM NUCLEAR DATA

```

```

PI= 3.1415927

```

```

DO 25 I=1,NUMEL
VD(I)=V(I)*D(I)
ZLMBDA(I)=ZNU*SGF(I)/SGA(I)-1.
VLMBDA(I)=V(I)*ZLMBDA(I)*SGA(I)
ZOMEGA(I)=V(I)*SGA(I)*ALPHA(I)*FISFAC*SGF(I)/HBAR
25 CONTINUE

```

```

18 FORMAT(5E15.4)
19 FORMAT(2E20.4,4G10.5)

```

```

C----- GEOMETRY OF SYSTEM NODAL POINTS -----

```

```

40 WRITE(6,40)
FORMAT('///1X,GEOMETRY OF SYSTEM NODAL POINTS'///)

```

```

41 READ(5,41) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)
FORMAT(5X,15,2F15.5)

```

```

60 WRITE(6,60) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)
FORMAT(1X,'SYSTEM NODAL POINT NO.',13,5X,'R COORD.= ',F10.6,5X,'
1Z COORD.= ',F10.6)

```

```

C*****
C CORRESPONDENCE TABLE BETWEEN SYSTEM AND ELEMENT NODAL POINTS
C*****
C*****@

```



```

C      WRITE(6,73)
C      FORMAT(/,2X,'CONNECTIVITY MATRIX',/, 'EL #',33X,'TYPE',/)
C      DO 75 I=1,NUMEL
C      READ(5,76) ELEMENT(I),ELNOD(I,1),ELNOD(I,2),ELNOD(I,3),ITYPE(I)
C      WRITE(6,77) ELEMENT(I),ELNOD(I,1),ELNOD(I,2),ELNOD(I,3),ITYPE(I)
C      75 CONTINUE
C      76 FORMAT (5I10)
C      77 FORMAT (2X,I3,4I10)
C
C      C CALCULATE GEOMETRY FOR EACH ELEMENT
C      C*****
C      WRITE(6,110)
C      110 FORMAT (1X,/,1X,'GEOMETRY CALCULATIONS FOR EACH ELEMENT')
C      C-----
C      C LOAD SYSTEM NODE COORDINATES (R,Z) INTO
C      C ELEMENT NODE COORDINATES (R1,R2,R3,Z1,Z2,Z3)
C
C      WRITE(6,95)
C      95 FORMAT (1X,/,1X,'ELEMENT',5X,'R1',9X,'Z1',9X,'R2',9X,'Z2',9X,'R3',
C      19X,'Z3',/)
C
C      DO 100 I=1,NUMEL
C
C      J=ELNOD(I,1)
C      R1(I)=R(J)
C      Z1(I)=Z(J)
C
C      K=ELNOD(I,2)
C      R2(I)=R(K)
C      Z2(I)=Z(K)
C
C      L=ELNOD(I,3)
C      R3(I)=R(L)
C      Z3(I)=Z(L)
C
C      WRITE      (5,105) I,R1(I),Z1(I),R2(I),Z2(I),R3(I),Z3(I)
C      100 CONTINUE
C      105 FORMAT (3X,I3,3X, 7(F10.6,1X))
C
C      COMPUTE A1,A2,A3,B1,B2,B3,AREA FOR EACH ELEMENT
C      C-----
C      WRITE      (6,120)

```



```
C      120 FORMAT ('X//IX,'ELEMENT','5X,'A1','9X,'A2','9X,'A3','9X,'B1','9X,'B2',  
C        '19X','B3','8X','AREA',//)  
  
C          DO 140 I=1, NUMEL  
C            A(I,1)=R3(I)-R2(1)  
C            A(I,2)=R1(1)-R3(1)  
C            A(I,3)=R2(1)-R1(1)  
C            B(1,1)=Z2(1)-Z3(1)  
C            B(1,2)=Z3(1)-Z1(1)  
C            B(1,3)=Z1(1)-Z2(1)  
C            AREA(1)=0.5*(A(1,1)+A(1,2)*B(1,1)+B(1,2)*B(1,3),  
C              WRITE(6,130) I,A(1,1),A(1,2),A(1,3),B(1,1),B(1,2),B(1,3),  
C                LAREA(1)  
C          140 CONTINUE  
  
C      130 FORMAT ('3X,I3,3X,    7(F12.7,1X))  
  
C----- ZERO THE BIGA,       BIGC AND BIGAB MATRICES-----  
  
C          DO 194 KK=1,NUMSNP  
C            DO 193 II=1,NUMSNP  
C              BIGA(KK,II)=0.  
C              PART(KK,II)=0.  
C              BIGAB(KK,II)=0.0  
C              BIGC(KK,II)=0.  
C            CONTINUE  
C          194 CONTINUE  
  
C*****  
  
C----- PEPSI CALCULATES THE BIGA MATRIX AND PART OF THE BIGAB -----  
  
C          CALL PEPST (NJMEL,NUMSNP,R1,R2,R3,AREA,ELNOD,  
C            1     BIGA,BIGAB,VLMBDA)  
  
C          WRITE(6,195)  
C          FORMAT('//5X,'3IGA MATRIX',)//  
C          WRITE(6,196)  
C          FORMAT('//5X,'BIGAB MATRIX FROM PEPST, NO DEL**2 CONTRIBUTION',//)
```



```

C-----
C CRUISE CALCULATES THE BIGAB MATRIX
C-----
C
C CALL CRUISE (NJMEL,NUMSNP, R1,R2,R3,AREA,ELNOD,
C 1 BIGAB, A,B,VD)
C
C WRITE(6,197)
C 197 FORMAT(/5X,'BIGAB MATRIX FROM CRUISE, WITH DEL**2 TERM',//)
C-----
C
C DO 255 I=1,NUPBP
C II=IUPBP(I)
C DO 251 J=1,NUMSNP
C BIGA(J,II)=0
C BIGA(II,J)=0
C BIGAB(J,II)=0
C BIGAB(II,J)=0
C 251 CONTINUE
C BIGA(II,II)=1
C BIGAB(II,II)=0
C 255 CONTINUE
C
C D1=1.
C CALL LINV3F(BIGA,B,1,NUMSNP,NUMSNP,D1,2,2,PART,IER)
C WRITE(6,250)
C 250 FORMAT(/5X,'3IGA INVERSE',//)
C-----
C
C WRITE OUT NUC-EAR DATA
C
C WRITE(6,3105) ZNJ,FISFAC,HBAR,EPSVAL,ERRVAL,AFUEL
C 3105 FORMAT(/5X,'NJCLEAR DATA',2X,'ZNU=',G12.6/2X,'FISFAC=',G12.6/2X,
C 1 'HBAR=',G12.6/2X,'EPSVAL=',G12.6/2X,'ERRVAL=',G12.6/2X,'AFUEL=',
C 2 G12.6)
C WRITE(6,3110)
C 3110 FORMAT(1H0,2X,'ELEM',6X,'D',8X,'SGA',9X,'SGF',8X,'ALPHA',9X,'VD',
C 1 7X,'ZLMBDA',7X,'VLMBDA',7X,'ZUMEGA',8X,'V',)
C DO 3115 I=1,NJMEL

```



```

WRITE(6,3120) (I,D(I),SGA(I),SGF(I),ALPHA(I),VD(I),ZLMBDA(I),
1 VLMBDA(I),ZOMEGA(I),V(I))
3115 CONTINUE
3120 FORMAT(2X,I4,10(1PE12.4))
C
C
NUMEQ=NUMSNP-NJPBP
C
EXTERNAL YVETTE
C
CALL YVET (NJMEL,NUMEQ,NUMSNP,
1 ITYPE,ELNOD,CM,BIGA,BIGAB,
2 PPSI,PV,NCOUNT,ZOMEGA,WP)
C----- INITIALIZE ARGUMENTS OF DVOGER -----
C
PTIME= 1.E-17
STAR=0.01
NIT=0
T=0
JSTART=0
EPS= EPSVAL
IER=0
C
WRITE(6,999) ITITLE
999 FORMAT(1H1,20A4)
C
WRITE(6,317)
WRITE(6,318) I, MTH, MAXDER, JSTART, H, HMIN, HMAX, EPS, NCOUNT
317 FORMAT (//10X, 'INITIAL ARGUMENTS',//)
318 FORMAT (2X, I=, G10.4, 2X, MTH=, I4, 2X, MAXDER=, I4, 2X, JSTART=, I4,
12X, H=, G10.4, 2X, HMIN=, G10.4, 2X, HMAX=, G10.4, 2X, EPS=, G10.4, 2X,
2, NCOUNT=, I2,/)
C
C
DO 320 I=1, NUMSNP
PSI(1,I)=PSIIV(I)
320 CONTINUE
C
DO 321 I=1, NUPBP
J=IUPBP(I)
PSI(1,J)=0.0
321 CONTINUE
C
NUMEQ=NUMSNP-NJPBP
C
C

```


C-----
 C CONSTRUCT THE 3IGC MATRIX ON THE ELEMENT LEVEL
 C-----

DO 200 L=1,NUMEL

DO 583 I=1,3

DO 582 J=1,3

DO 581 K=1,3

CM(L,I,J,K)=0.

CONTINUE

CONTINUE

LL=ITYPE(L)

IF (LL.NE. 0) GO TO 200

CC=PI*AREA(L)/180.

CM(L,3,3,3)=CC*(6.0*RI(L)+6.0*RR2(L)+24.*R3(L))
 CM(L,3,3,2)=CC*(2.0*RI(L)+4.0*RR2(L)+6.*R3(L))
 CM(L,3,3,1)=CC*(4.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,3,2,3)=CC*(2.0*RI(L)+4.0*RR2(L)+6.0*R3(L))
 CM(L,3,2,2)=CC*(2.0*RI(L)+2.0*RR2(L)+3.0*R3(L))
 CM(L,3,2,1)=CC*(2.0*RI(L)+2.0*RR2(L)+3.0*R3(L))
 CM(L,3,1,3)=CC*(2.0*RI(L)+2.0*RR2(L)+3.0*R3(L))
 CM(L,3,1,2)=CC*(2.0*RI(L)+2.0*RR2(L)+3.0*R3(L))
 CM(L,3,1,1)=CC*(2.0*RI(L)+2.0*RR2(L)+3.0*R3(L))
 CM(L,2,3,3)=CC*(2.0*RI(L)+6.0*RR2(L)+4.0*R3(L))
 CM(L,2,3,2)=CC*(2.0*RI(L)+6.0*RR2(L)+4.0*R3(L))
 CM(L,2,3,1)=CC*(2.0*RI(L)+6.0*RR2(L)+4.0*R3(L))
 CM(L,2,2,3)=CC*(6.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,2,2,2)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,2,2,1)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,2,1,3)=CC*(6.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,2,1,2)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,2,1,1)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,3,3)=CC*(6.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,1,3,2)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,3,1)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,2,3)=CC*(6.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,1,2,2)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,2,1)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,1,3)=CC*(6.0*RI(L)+2.0*RR2(L)+6.0*R3(L))
 CM(L,1,1,2)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))
 CM(L,1,1,1)=CC*(2.0*RI(L)+2.0*RR2(L)+2.0*R3(L))

200 CONTINUE

C


```

C 351 CONTINUE
C IF (NCOUNT.EQ. 1) GO TO 312
C
DO 210 L=1,NUMEL
LL=ITYPE(L)
IF(LL.NE. 0) GO TO 210
N1=ELNOD(L,1)
N2=ELNOD(L,2)
N3=ELNOD(L,3)
DO 88 J=1,3
DO 90 N=1,3
ALFA(N,J)=PSIIV(N1)*CM(L,N,J,1)+PSIIV(N2)*CM(L,N,J,2)
+PSIIV(N3)*CM(L,N,J,3)
1
90 CONTINUE
88 CONTINUE
DO 150 K=1,3
KK=ELNOD(L,K)
DO 180 I=1,3
II=ELNOD(L,I)
BIGC(KK,II)=BIGC(KK,II)+ALFA(K,I)*ZOMEGA(L)
180 CONTINUE
150 CONTINUE
210 CONTINUE
C
C INSERTION OF BOUNDARY POINTS
DO 285 I=1,NUPBP
II=IUPBP(I)
DO 291 J=1,NUMSNP
BIGC(J,II)=0
BIGC(II,J)=0
291 CONTINUE
285 CONTINUE
C
DO 310 I=1,NUMSNP
DO 310 J=1,NUMSNP
BIGCC(I,J)=0
PART(I,J)=0
DO 300 K=1,NUMSNP
BIGCC(I,J)=BIGCC(I,J)+(BIGAB(K,J)-BIGC(K,J))*BIGA(I,K)
PART(I,J)=PART(I,J)+(BIGAB(K,J)-2.*BIGC(K,J))*BIGA(I,K)
300 CONTINUE
310 CONTINUE
C
C

```



```

I1=I+NUF
I2=I+2*NUF
I3=I+3*NUF
I4=I+4*NUF
WRITE(6,356)(I,S(I),I1,S(I1),I2,S(I2),I3,S(I3),I4,S(I4))
359 CONTINUE
362 CONTINUE
C
DO 27 MN=1,NUMSNP
PSIIV(MN)=PSI(I,MN)
DO 28 I=1,NUMSNP
BIGC(MN,I)=.0
28 CONTINUE
27 CONTINUE
C IF (T .LT. TF) GO TO 351
C
C STOP
END

```



```

C SUBROUTINE PEPSI (NUMEL,NUMSNP,R1,R2,R3,AREA,ELNOD,
C   BIGA,BIGAB,VLMBDA)
C *****
C PEPSI CALCULATES THE BIGA MATRIX AND PART OF THE BIGAB
C *****
C INTEGER*4 ELNOD
C
C   DIMENSION R1(NJMEL),R2(NUMEL),R3(NUMEL),BIGA(NUMSNP,NUMSNP),
C   1   BIGAB(NUMSNP,NUMSNP),AMATRIX(3,3),
C   2   VLMBDA(NUMEL),AREA(NUMEL),ELNOD(NUMEL,3)
C
C   PI=3.1415927
C
C   CALCULATE THE 3X3 D(I,J) MATRIX FOR ELEMENT L
C
C   DO 200 L=1,NUMEL
C     COEFFA=(PI/30.0)*AREA(L)
C
C     AMATRIX(1,1)=COEFFA   *(6.0*R1(L)+2.0*R2(L)+2.0*R3(L))
C     AMATRIX(1,2)=COEFFA   *(2.0*R1(L)+2.0*R2(L)+R3(L))
C     AMATRIX(1,3)=COEFFA   *(2.0*R1(L)+R2(L)+2.0*R3(L))
C     AMATRIX(2,1)=AMATRIX(1,2)
C     AMATRIX(2,2)=COEFFA   *(2.0*R1(L)+6.0*R2(L)+2.0*R3(L))
C     AMATRIX(2,3)=COEFFA   *(R1(L)+2.0*R2(L)+2.0*R3(L))
C     AMATRIX(3,1)=AMATRIX(1,3)
C     AMATRIX(3,2)=AMATRIX(2,3)
C     AMATRIX(3,3)=COEFFA   *(2.0*R1(L)+2.0*R2(L)+6.0*R3(L))
C
C-----
C STORE ELEMENT MATRIX, AMATRIX, INTO SYSTEM MATRIX, BIGA
C-----
C
C   DO 20 K=1,3
C     KK=ELNOD(L,K)
C     DO 10 I=1,3
C       II=ELNOD(L,I)
C       BIGA(KK,II)=BIGA(KK,II)+AMATRIX(K,I)
C       BIGAB(KK,II)=BIGAB(KK,II)+VLMBDA(L)*AMATRIX(K,I)
C     10 CONTINUE
C   20 CONTINUE
C
C   200 CONTINUE
C     RETURN
C   END

```



```

C SUBROUTINE CRJISE (NUMEL, NUMSNP,      R1, R2, R3, AREA, ELNOD, BIGAB,
C   A, B, VD)
C   *
C CRUISE CALCULATES THE BIGAB MATRIX
C-----
C INTEGER*4 ELNOD
C
C DIMENSION R1(NJMELE), R2(NJMELE), R3(NJMELE), AREA(NUMEL),
C   1 BIGAB(NUMSNP, NUMSNP), BMATRIX(3, 3),
C   2 ELNOD(NUMEL, 3), A(NUMEL, 3), B(NUMEL, 3), VD(NUMEL)
C
C PI=3.1415927
C-----
C CALCULATE THE 3X3 B(I, J) MATRIX FOR ELEMENT L
C-----
C DO 200 L=1, NUMEL
C   COEFFB=PI*VD(L)/(6.0*AREA(L))
C
C   BMATRIX(1,1)=COEFFB *(-R1(L)*(2.0*B(L,1)**2+2.0*A(L,1)**2)-R2(L)*
C     1(B(L,1)*B(L,2)+B(L,1)**2+A(L,1)*A(L,1)**2)-R3(L)*(B(L,1)*
C     2B(L,3)+B(L,1)*A(L,3)+A(L,1)**2)+2.0*AREA(L)*B(L,1))
C
C   BMATRIX(1,2)=COEFFB *(-R1(L)*(2.0*B(L,1)*B(L,2)+2.0*A(L,1)*
C     1A(L,2))-R2(L)*(B(L,2)**2+B(L,1)*B(L,2)+A(L,2)**2+A(L,1)*A(L,2))-
C     2B(L,2)*B(L,3)+B(L,1)*B(L,3)+A(L,2)*A(L,3)+A(L,1)*A(L,2))+
C     32.0*AREA(L)*B(L,2))
C
C   BMATRIX(1,3)=COEFFB *(-R1(L)*(2.0*B(L,1)*B(L,3)+2.0*A(L,1)*
C     1A(L,3))-R2(L)*(B(L,2)*B(L,3)+B(L,1)*B(L,3)+A(L,2)*A(L,3)+A(L,1)*
C     2A(L,3))-R3(L)*(B(L,3)**2+B(L,1)*B(L,3)+A(L,3)**2+A(L,1)*A(L,3))+
C     32.0*AREA(L)*B(L,3))
C
C   BMATRIX(2,1)=BMATRIX(1,2)
C
C   BMATRIX(2,2)=COEFFB *(-R1(L)*(B(L,2)**2+B(L,1)*B(L,2)+A(L,2)**2+
C     1A(L,1)*A(L,2))-R2(L)*(2.0*B(L,2)**2+2.0*A(L,2)**2)-R3(L)*(B(L,3)*
C     2B(L,2)+B(L,2)*A(L,3)*A(L,2)+A(L,2)**2)+2.0*AREA(L)*B(L,2))
C
C   BMATRIX(2,3)=COEFFB *(-R1(L)*(B(L,2)*B(L,3)+B(L,1)*B(L,3)+A(L,2)*
C     1A(L,3)+A(L,1)*A(L,3))-R2(L)*(2.0*B(L,2)*B(L,3)+2.0*A(L,2)*
C     2B(L,3)+B(L,3)*A(L,3)+A(L,2)*A(L,3)+A(L,1)*A(L,3))-
C     32.0*AREA(L)*B(L,3))
C

```



```

C      BMATRX(3,1)=BMATRX(1,3)
C      BMATRX(3,2)=BMATRX(2,3)
C      BMATRX(3,3)=COEFFB  *(-R1(L)*(B(L,3)**2+B(L,1)*B(L,3)+A(L,3)**2+
1A(L,1)*A(L,3))-R2(L)*(B(L,3)**2+B(L,2)*B(L,3)+A(L,3)**2+A(L,2)*
2A(L,3))-R3(L)*(2.0*B(L,3)**2+2.0*A(L,3)**2)+2.0*AREA(L)*B(L,3))
C
C      STORE ELEMENT MATRIX, AMATRX, INTO SYSTEM MATRIX, BIGA
C-----
C      DO 20 K=1,3
C      KK=ELNOD(L,K)
C      DO 10 I=1,3
C      II=ELNOD(L,I)
C      BIGAB(KK,II)=BIGAB(KK,II)+BMATRX(K,I)
C      10 CONTINUE
C      20 CONTINUE
C      200 CONTINUE
C      RETURN
C      END

```



```

SUBROUTINE YVET(NUMEL,NUMEQ,NUMSNP, BIGCC,PART,NWK,PSI,PW,DPSI,
1 ITYPE,ELNDD,CM,BIGA,BIGAB,
2 PDPSI,PV,NCOJNT,ZOMEGA,WP)
      IUPBP,PSIIV,NUPBP,BIGC,
      BIGCC,PART,NWK,PSI,PW,DPSI,
      PDPSI,PV,NCOJNT,ZOMEGA,WP)

C
C
      INTEGER*4 ELNDD
C
      DIMENSION PART(132,132),BIGA(132,132),BIGAB(132,132),BIGC(132,132)
1    BIGCC(132,132),ALFA(3,3),ITYPE(220),ELNDD(220,3),IUPBP(221)
5    ZOMEGA(220),WP(132,132),CM(220,3,3),DPSI(132)
6    PSIIV(132),PDPSI(132),PW(20000),PSI(8,132),PV(132)
      PI=3.1415927
      RETURN
C
C*****
C
      ENTRY YVETTE(PSI,T,MQ,DPSI,WP,IND)
C
C YVETTE IS AN EXTERNAL SUBROUTINE REQUIRED BY SUBROUTINE DVJGER
C
C
      DO 15 I=1,NUMSNP
        PV(I)=PSI(1,I)
15      CONTINUE
      DO 50 I=1,NUPBP
        J=IUPBP(I)
        PV(J)=0.
50      CONTINUE
      IF (NCOJNT.NE.1) GO TO 05
C
      DO 27 MM=1,NUMSNP
        PSIIV(MM)=PV(MM)
      DO 28 I=1,NUMSNP
        BIGC(MM,I)=0.
28      CONTINUE
      DO 27 MM=1,NUMSNP
        PSIIV(MM)=PV(MM)
      DO 28 I=1,NUMSNP
        BIGC(MM,I)=0.
28      CONTINUE
      DO 210 L=1,NUMEL
        IF(LL.NE.0) GO TO 210
        LL=ITYPE(L)
        N1=ELNDD(L,1)
        N2=ELNDD(L,2)
        N3=ELNDD(L,3)
        DO 86 J=1,3
          DO 90 N=1,3
            ALFA(N,J)=PSIIV(N1)*CM(L,J,1)+PSIIV(N2)*CM(L,N,J,2)
            +PSIIV(N3)*CM(L,N,J,3)
          1 CONTINUE
90      CONTINUE

```

240
241
242
243
244
245
246
201
202
203
204
205
206
207
208
209
210
211


```

      86 CONTINUE
      DO 150 K=1,3
      KK=ELNOD(L,K)
      DO 280 I=1,3
      II=ELNOD(L,I)

      BIGC(KK,II)=BIGC(KK,II)+ALFA(K,I)*ZOMEGA(L)

      C
      280 CONTINUE
      150 CONTINUE
      210 CONTINUE

      C
      INSRTION OF BOUNDARY POINTS

      DO 285 I=1,NUP3P
      II=IUPBP(I)
      DO 281 J=1,NUMSNP
      BIGC(J,II)=0.
      BIGC(II,J)=0.
      C
      281 CONTINUE
      285 CONTINUE

      DO 311 I=1,NUMSNP
      DO 310 J=1,NUMSNP
      BIGC(I,J)=0.
      PART(I,J)=0.
      DO 300 K=1,NUMSNP
      BIGC(I,J)=BIGC(I,J)+(BIGAB(K,J)-BIGC(K,J))*BIGA(I,K)
      PART(I,J)=PART(I,J)+(BIGAB(K,J)-2.*BIGC(K,J))*BIGA(I,K)
      C
      300 CONTINUE
      310 CONTINUE
      311 CONTINUE

      C
      05 CONTINUE

      C
      IF (IND.EQ. 1) GO TO 205

      C
      DO 81 K=1,NUMSNP
      DPSI(K)=0.
      IF(K.GT. NUMEQ)GO TO 81
      DO 80 I=1,NUMSNP
      DPSI(K)=DPSI(K)+BIGC(K,I)*PV(I)
      C
      80 CONTINUE
      81 CONTINUE
      IF(T.GT.0.0) RETURN
      WRITE(6,60)
      FORMAT(2X,'FROM YVETTE',5X,'I=',G20.10)
      60 WRITE(6,65)((DPSI(K),K=1,NUMSNP)
      WRITE(6,65)(PSI(1,K),K=1,NUMSNP)
      FORMAT(10(1X,E11.5))
      65

```



```

C
C 205
C
C      RETURN
C      CONTINUE
C      WHEN MTH = 1 COMPUTE THE JACOBIAN
C
C      DO 900 K=1,NUMSNP
C      DO 880 L=1,NUMSNP
C      WP(K,L)=PART(K,L)
C      CONTINUE
C      CONTINUE
C      RETURN
C      END
C
C 880
C 900

```



```

C----- DECLARATION STATEMENTS -----
C
C      IMPLICIT REAL*4 (A-H,O-Z), INTEGER*4 (I-N)
C
C      INTEGER*2 NAME
C      INTEGER*4 SYSNJD,ELEMNT,ELNOD,RHO
C
C      COMMON/NUMB/ELNOD(220,3),NUMEL,NUMSNP,NFULEL,NELROW,NBP,IBP(30)
C      1,NPROB,NTYPE,ITYPE(220)
C      COMMON/CMCL/BAGA(132,7),BAGAB(132,7),SCRAPC(132,28),NDE,NCMPK,
C      1,NAME(132,7)
C
C      COMMON/SOLVE/PI,VDF,VDC,VLWBDA,ZOMEGA,VSMAC,SCALE
C
C      COMMON R1(220),R2(220),R3(220),A(220,3),B(220,3),AREA(220),Z1(220)
C      1,Z2(220),Z3(220),A12(220),A23(220),A31(220),GAMMA(220),DELTA(220)
C
C      DIMENSION PSTARI(132),R(132),Z(132),TPLOT(20)
C      DIMENSION PSI(8,132),ERROR(132),YMAX(132)
C      DIMENSION SYSNJD(132),ELEMNT(220),TITLE(20)
C      DIMENSION Y(7,132),WS(1430)
C      DIMENSION ICORE(70)
C
C      CALL ERRSET(207,256,0,1,1,209)
C
C
C      READ(5,998) TITLE
C      WRITE(6,999) TITLE
C      FORMAT(20A4)
C      998
C      999      FORMAT(1H1,18X,20A4)
C
C      READ(5,10) NJMEL,NBP,NUMSNP,NFULEL,NELROW,MXEVAL,NCOUNT,NCMPK
C      READ(5,10) NDE,NL
C      WRITE(6,997) NDE,NL
C      997      FORMAT(/,2X,'NDE=',110/2X,'NL=',110)
C      READ(5,200)(IBP(I),I=1,NBP)
C      10      READ(8110)
C      200      FORMAT(16I5)
C
C      *****
C      GENERAL PROBLEM DATA
C      *****
C      WRITE(6,11) NUMEL, NBP,NUMSNP,NFULEL,NELROW,MXEVAL,NCOUNT,NCMPK

```



```

11 FORMAT(/2X,'# OF ELEMENTS=',I5//2X,'# OF SYSTEM BOUNDARY POINTS=
1,I5//2X,'# OF SYSTEM NODAL POINTS=',I5//2X,'# OF FUEL ELEMENTS=',
2,I5//2X,'# OF ELEMENTS IN A ROW=',I5//2X,'MAX. # OF EVALUATIONS=',I
3,I5//2X,'NCOUNT=',I5//2X,'NCMPK=',I5)
WRITE(6,10) (I3P(I),I=1,NBP)

C
READ(5,15) VELOC, DSUBF, DSUBC, SGMAAF, SGMAAC, SGMAF, FISFAC, HBAR,
READ(5,15) ZNJ, ALPHA, RMSEPS, AFUEL, TSTART, TEND, PINITY, SCALE
READ(5,15) HMIN, HMAX
WRITE(6,19) HMIN, HMAX
19 FORMAT(/2X,'HMIN=',G12.6/2X,'HMAX=',G12.6)

C
IF(TSTART.GT.0.) READ(5,13) (PSTART(I),I=1,NUMSNP)
IF(TSTART.GT.0.) WRITE(6,13) (PSTART(I),I=1,NUMSNP)
13 FORMAT(5G15.5)

C
READ(5,10) MTH, MAXDER, NPROB, NTYPE, JPLOT, NPLOT, IRUN
WRITE(5,12) NPROB, NTYPE, JPLOT, NPLOT, IRUN
FORMAT(/2X,'NPROB=',I5/2X,'NTYPE=',I5/2X,'JPLOT=',I5/2X,'NPLOT=',I
15/2X,'IRUN=',I5)
IF(JPLOT.EQ.1) READ(5,13) (TPLOT(I),I=1,NPLOT)
IF(JPLOT.EQ.1) WRITE(6,13) (TPLOT(I),I=1,NPLOT)
IF(JPLOT.EQ.1) ITPC=1

C
WRITE(6,16) VELOC, DSUBF, DSUBC, SGMAAF, SGMAAC, SGMAF, FISFAC, HBAR, ZNU,
1 ALPHA, RMSEPS, AFUEL, TSTART, TEND, PINITY, SCALE
16 FORMAT(/75X,'NUCLEAR DATA',/2X,'VELOC=',G12.6/2X,'DSUBF=',G12.6/2X,
1,'DSUBC=',G12.5/2X,'SGMAAF=',G12.6/2X,'SGMAAC=',G12.6/2X,'SGMAF=',
2,G12.6/2X,'FISFAC=',G12.6/2X,'HBAR=',G12.6/2X,'ZNU=',G12.6/2X,
3,'ALPHA=',G12.6/2X,'RMSEPS=',G12.6/2X,'AFUEL=',G12.6/2X/2X,
4,'TSTART=',G12.6/2X,'TEND=',G12.6/2X,'PINITY=',G12.6/2X,
5,'SCALE=',E10.5)

C
15 FORMAT(8G10.5)

C
CALCULATE PHYSICAL CONSTANTS FROM NUCLEAR DATA

C
ICOUNT=0
ZLMBDA=ZNU*SGMAF/SGMAAF-1.
VDF=VELOC*DSUBF
VDC=VELOC*DSUBC
VLMBDA=VELOC*ZLMBDA*SGMAAF
ZOMEGA=2.*VELOC*SGMAAF*ALPHA*FISFAC*SGMAF/(HBAR*(AFUEL**2.))*SCAL
1E
IF(NTYPE.EQ.0) ZOMEGA=ZOMEGA*(AFUEL**2)/2.
VSMAC=VELOC*SGMAAC
PI=3.1415927

```



```

WRITE(6,17)ZLMBDA,VDF,VDC,VLMBDA,ZOMEGA,VSMAC,VELOCIT
17 FORMAT(/,5X,'PHYSICAL CONSTANTS',/2X,'ZLMBDA=',G12.6/2X,'VDF=',
1G12.6/2X,'VDC=',G12.6/2X,'VLMBDA=',G12.6/2X,'ZOMEGA=',G12.6/2X,
2'VSMAC=',G12.6/2X,'VELOCIT=',G12.6)

```

C C

```

DO 50 I=1,NUMSNP
Y(1,I)=0.
Y(2,I)=0.
50 CONTINUE

```

Y(1,1)=PINITY

C C C C C C C

NODAL NEIGHBOR CONNECTIVITY

```

DO 18 I=1,NUMSNP
READ(5,111)(NAME(I,J),J=1,NCMPK)
WRITE(6,111) (NAME(I,J),J=1,NCMPK)
111 CONTINUE
DO 172 KK=1,NCMPK
172 FORMAT(9(2X,I3))
DO 171 I=1,NCMPK
171 BAGA(KK,I)=0.
BAGAB(KK,I)=0.
DO 174 J=1,NCMPK
JJ=(I-1)*(2*NCMPK-I)/2+J
SCRAPC(KK,JJ)=0.
174 CONTINUE
171 CONTINUE
172 CONTINUE

```

C

```

WRITE(6,8)
8 FORMAT('1','VJCLEAR DATA',///)

```

C C C

NSNPSQ= NUMSNP * NUMSNP

----- GEOMETRY OF SYSTEM NODAL POINTS -----

C C

```

WRITE(6,40)
40 FORMAT (///1X,'GEOMETRY OF SYSTEM NODAL POINTS',///)

```

READ(5,41) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)


```

41  FORMAT(5X,I5,2F15.5)
C
  WRITE      (6,60) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)
60  FORMAT (1X,'SYSTEM NODAL POINT NO.',I3,5X,'R COORD.= ',F10.6,5X,'
      1Z COORD.= ',F10.6)
C
  IF(JPLGT,EQ,0) GO TO 72
  WRITE(7,7000) IRUN,HBAR,NUMSNP
7000  FORMAT(1X,'RUN NUMBER=',I3,F10.5,I5)
  WRITE(7,7001) (R(I),Z(I),I=1,NUMSNP)
7001  FORMAT(4E15.6)
C
C *****
C CURRESPONDENCE TABLE BETWEEN SYSTEM AND ELEMENT NODAL POINTS
C *****
72  WRITE(6,73)
73  FORMAT(7I2X,'CONNECTIVITY MATRIX',//,' EL #',33X,'TYPE'//)
  DO 75 I=1,NUMEL
  READ(5,76) ELEMENT(I),ELNOD(I,1),ELNOD(I,2),ELNOD(I,3),ITYPE(I)
  WRITE(6,77) ELEMENT(I),ELNOD(I,1),ELNOD(I,2),ELNOD(I,3),ITYPE(I)
75  CONTINUE
76  FORMAT (5I10)
77  FORMAT (2X,I3,4I10)
C
C *****
C CALCULATE GEOMETRY FOR EACH ELEMENT
C *****
  WRITE(6,I10)
110  FORMAT (1X//1X,'GEOMETRY CALCULATIONS FOR EACH ELEMENT')
C
-----
C LOAD SYSTEM NODE COORDINATES (R,Z) INTO
C ELEMENT NODE COORDINATES (R1,R2,R3,Z1,Z2,Z3)
C
-----
  WRITE(6,95)
95  FORMAT (1X//1X,'ELEMENT',5X,'R1',9X,'Z1',9X,'R2',9X,'Z2',9X,'R3',
      19X,'Z3',/)
  DO 100 I=1,NUMEL
  J=ELNOD(I,1)
  R1(I)=R(J)
  Z1(I)=Z(J)

```



```

C      K=ELNOD(I,2)
C      R2(I)=R(K)
C      Z2(I)=Z(K)
C
C      L=ELNOD(I,3)
C      R3(I)=R(L)
C      Z3(I)=Z(L)
C
C      105 WRITE (6,105) I,R1(I),Z1(I),R2(I),Z2(I),R3(I),Z3(I)
C      106 FORMAT (1X,2X,13,4X,7(F10.6,1X))
C      100 CONTINUE
C
C-----
C      COMPUTE A1,A2,A3,B1,B2,B3,AREA FOR EACH ELEMENT
C-----
C
C      120 WRITE (6,120)
C      121 FORMAT (1X//1X,'ELEMENT',5X,'A1',9X,'A2',9X,'A3',9X,'B1',9X,'B2',
C      122 19X,'B3',8X,'AREA',//)
C
C      DO 140 I=1,NUMEL
C      A(1,1)=R3(I)-R2(I)
C      A(1,2)=R1(I)-R3(I)
C      A(1,3)=R2(I)-R1(I)
C      B(1,1)=Z2(I)-Z3(I)
C      B(1,2)=Z3(I)-Z1(I)
C      B(1,3)=Z1(I)-Z2(I)
C      AREA(I)=0.5*(A(1,2)*B(1,1)-A(1,1)*B(1,2))
C
C      140 WRITE(6,130) I,A(1,1),A(1,2),A(1,3),B(1,1),B(1,2),B(1,3),
C      141 1AREA(I)
C      140 CONTINUE
C
C      130 FORMAT (1X,2X,13,4X,7(F10.6,1X))
C
C-----
C      COMPUTE A12,A23,A31,GAMMA,DELTA FOR EACH ELEMENT
C-----
C
C      150 WRITE (6,150)
C      151 FORMAT ('1','ELEMENT',5X,'A12',8X,'A23',8X,'A31',6X,'GAMMA',7X,'DE
C      152 1LTA',//)
C
C      DO 170 L=1,NUMEL
C      A12(L)=0.5* (R1(L)*Z2(L)-R2(L)*Z1(L))
C      A23(L)=0.5* (R2(L)*Z3(L)-R3(L)*Z2(L))

```


C

```

JSKF=0
WRITE(6,317) TSTART,TEND,MAXDER,JSKF,HMIN,HMAX,RMSEPS
317 FORMAT(/5X,'INITIAL ARGUMENTS',/)
318 FORMAT(2X,'TSTART=',G12.6/2X,'TEND=',G12.6/2X,'MAXDER=',I5/2X,
1,JSKF=',I5/2X,'HMIN=',G12.5/2X,'HMAX=',G12.6/2X,'RMSEPS=',G12.6)
C
H = HMIN*1000.
CALL SDESOL(Y,I,L,TSTART,TEND,NDE,NL,NDE,JSKF,MAXDER,I,H,HMIN,HMAX,
1,RMSEPS,WS)
9998 WRITE(6,9997) JSKF
9997 FORMAT(/2X,'JSKF=',I5)
C
9999 STOP
END

```


C

```

DO 7050 K=1,3
KK=ELNOD(L,K)
DO 7040 I=1,3
II=ELNOD(L,I)
DO 7035 M=1,NCMPK
KKM=NAME(KK,M)
IF(KKM.EQ.II) GO TO 7032
CONTINUE
7035 BAGAB(KK, M)=BAGAB(KK, M)+CC*AMATRIX(K,I)
7032 BAGAB(KK, M)=BAGAB(KK, M)+AMATRIX(K,I)
7040 CONTINUE
7050 CONTINUE
200 CONTINUE
8074 FORMAT(1P3E15.4)
WRITE(6,8071)
FORMAT(/2X,'BAGAB MATRIX FROM PEPSI')
WRITE(6,8072)((BAGA(I,J),J=1,NCMPK),I=1,NUMSNP)
WRITE(6,8073)
WRITE(6,8076)((BAGA(I,J),J=1,NCMPK),I=1,NUMSNP)
FORMAT(1P7E11.4)
8076 FORMAT(/2X,'BAGAB MATRIX FROM PEPSI')
8073 WRITE(6,8072)((BAGAB(I,J),J=1,NCMPK),I=1,NUMSNP)
8072 FORMAT(1P7E15.4)
C
RETURN
END

```



```

C      BMATRX(2,2)=COEFFB      *(-R1(L)*(B(L,2)**2+B(L,1)*B(L,2)+A(L,2)**2+
1A(L,1)*A(L,2))-R2(L)*(2.0*B(L,2)**2+2.0*A(L,2)**2)-R3(L)*(B(L,3)*
2B(L,2)+B(L,2)*A(L,3)*A(L,2)+A(L,2)**2)+2.0*AREA(L)*B(L,2))
C      BMATRX(2,3)=COEFFB      *(-R1(L)*(B(L,2)*B(L,3)+B(L,1)*B(L,3)+A(L,2)*
1A(L,3)+A(L,1)*A(L,3))-R2(L)*(2.0*B(L,2)*B(L,3)+2.0*A(L,2)*A(L,3))-
2R3(L)*(B(L,3)**2+B(L,2)*B(L,3)+A(L,3)**2+A(L,2)*A(L,3))+
32.0*AREA(L)*B(L,3))
C      BMATRX(3,1)=BMATRX(1,3)
C      BMATRX(3,2)=BMATRX(2,3)
C      BMATRX(3,3)=COEFFB      *(-R1(L)*(B(L,3)**2+B(L,1)*B(L,3)+A(L,3)**2+
1A(L,1)*A(L,3))-R2(L)*(B(L,3)**2+B(L,2)*B(L,3)+A(L,3)**2)+2.0*AREA(L)*B(L,3))
2A(L,3))-R3(L)*(2.0*B(L,3)**2+2.0*A(L,3)**2)+2.0*AREA(L)*B(L,3))
C      WRITE(6,35) ((I3MATRX(I,J),J=1,3),I=1,3)
C      35  FORMAT(2X,3E20.10)
C
C
C

```

```

DO 7050 K=1,3
KK=ELNDD(L,K)
DO 7040 I=1,3
II=ELNDD(I,I)
DO 7035 M=1,NCMPK
KKM=NAME(KK,M)
IF(KKM.EQ.II) GO TO 7032
CONTINUE
      BAGAB(KK, M)=BAGAB(KK, M)+CC*BMATRX(K,I)
7035 CONTINUE
7032 CONTINUE
7040 CONTINUE
7050 CONTINUE
8074 FORMAT(1P3E15.4)
8073 WRITE(6,8073) BAGAB_MATRIX FROM CRUISE'
      FORMAT(/2X,'BAGAB_MATRIX FROM CRUISE',I=1,NUMSNP)
8072 WRITE(6,8072)((BAGAB(I,J),J=1,NCMPK),I=1,NUMSNP)
      FORMAT(1P7E15.4)
C 8076 WRITE(7,8076)((BAGAB(I,J),J=1,NCMPK),I=1,NUMSNP)
      FORMAT(1P7E11.4)
C
C      RETURN
C      END

```



```

SUBROUTINE PSISQ
  THIS SUBROUTINE CALCULATES THE NON LINEAR PSI SQUARE TERM

  INTEGER*2 NAME
  INTEGER*4 SYSNJD, ELEMNT, ELNOD, RHO

  COMMON/NUMB/ELNOD(220,3), NJMEL, NUMSNP, VFULEL, NELROW, NBP, IBP(30)
  1, NPROB, NTYPE, I1TYPE(220)
  COMMON/CMCL/BAGA(132,7), BAGAB(132,7), SCRAPC(132,28), NDE, NCMPK,
  1 NAME(132,7)
  COMMON/SOLVE/PI,VDF,VDC,VLMBDA,ZOMEGA,VSMAC,SCALE

  COMMON R1(220),R2(220),R3(220),A(220,3),B(220,3),AREA(220),Z1(220)
  1,Z2(220),Z3(220),A12(220),A23(220),A31(220),GAMMA(220),DELTA(220)

  DIMENSION CMATRIX(3,9),CM(3,3,3)

  NCC=(NCMPK+1)*NCMPK/2
  DO 200 L=1,NJMEL
    LL=I1TYPE(L)
    IF(LL.NE.0) GO TO 200

    CC=PI*AREA(L)/180.

    CMATRIX(1,1)=CC*(24.*R1(L)+6.0*R2(L)+6.0*R3(L))
    CMATRIX(1,2)=CC*(6.0*R1(L)+4.0*R2(L)+2.0*R3(L))
    CMATRIX(1,3)=CC*(6.0*R1(L)+2.0*R2(L)+4.0*R3(L))
    CMATRIX(1,4)=CMATRIX(1,2)
    CMATRIX(1,5)=CC*(4.0*R1(L)+6.0*R2(L)+2.0*R3(L))
    CMATRIX(1,6)=CC*(2.0*R1(L)+2.0*R2(L)+2.0*R3(L))
    CMATRIX(1,7)=CMATRIX(1,3)
    CMATRIX(1,8)=CMATRIX(1,6)
    CMATRIX(1,9)=CC*(4.0*R1(L)+2.0*R2(L)+6.0*R3(L))
    CMATRIX(2,1)=CMATRIX(1,2)
    CMATRIX(2,2)=CMATRIX(1,5)
    CMATRIX(2,3)=CMATRIX(1,6)
    CMATRIX(2,4)=CMATRIX(1,9)
    CMATRIX(2,5)=CC*(6.0*R1(L)+24.*R2(L)+6.0*R3(L))
    CMATRIX(2,6)=CC*(2.0*R1(L)+6.0*R2(L)+4.0*R3(L))
    CMATRIX(2,7)=CMATRIX(1,6)
    CMATRIX(2,8)=CMATRIX(2,6)
    CMATRIX(2,9)=CC*(2.0*R1(L)+4.0*R2(L)+6.0*R3(L))
    CMATRIX(3,1)=CMATRIX(1,3)
    CMATRIX(3,2)=CMATRIX(1,6)
    CMATRIX(3,3)=CMATRIX(1,9)
  200 CONTINUE

```



```

C      CMATRX(3,4)=CMATRX(1,6)
C      CMATRX(3,5)=CMATRX(2,6)
C      CMATRX(3,6)=CMATRX(2,9)
C      CMATRX(3,7)=CMATRX(1,9)
C      CMATRX(3,8)=CMATRX(2,9)
C      CMATRX(3,9)=CC*(6.0*R1(L)+6.0*R2(L)+24.*R3(L))
C
C      IF(L.EQ.1) WRITE(6,888) ((I,J,CMATRX(I,J),J=1,9),I=1,3)
C      IF(L.EQ.1) WRITE(6,888) ((I,J,CMATRX(I,J),J=1,9),I=1,3)
C      888 FORMAT(2X,'I=',I5,5X,'J=',I5,5X,'CMATRX=',IPE12.4)
C
C      CM(1,1,1)=CMATRX(1,1)
C      CM(1,1,2)=CMATRX(1,2)
C      CM(1,1,3)=CMATRX(1,3)
C      CM(1,2,1)=CMATRX(1,4)
C      CM(1,2,2)=CMATRX(1,5)
C      CM(1,2,3)=CMATRX(1,6)
C      CM(1,3,1)=CMATRX(1,7)
C      CM(1,3,2)=CMATRX(1,8)
C      CM(1,3,3)=CMATRX(1,9)
C      CM(2,1,1)=CMATRX(2,1)
C      CM(2,1,2)=CMATRX(2,2)
C      CM(2,1,3)=CMATRX(2,3)
C      CM(2,2,1)=CMATRX(2,4)
C      CM(2,2,2)=CMATRX(2,5)
C      CM(2,2,3)=CMATRX(2,6)
C      CM(2,3,1)=CMATRX(2,7)
C      CM(2,3,2)=CMATRX(2,8)
C      CM(2,3,3)=CMATRX(2,9)
C      CM(3,1,1)=CMATRX(3,1)
C      CM(3,1,2)=CMATRX(3,2)
C      CM(3,1,3)=CMATRX(3,3)
C      CM(3,2,1)=CMATRX(3,4)
C      CM(3,2,2)=CMATRX(3,5)
C      CM(3,2,3)=CMATRX(3,6)
C      CM(3,3,1)=CMATRX(3,7)
C      CM(3,3,2)=CMATRX(3,8)
C      CM(3,3,3)=CMATRX(3,9)
C
C      IF(L.EQ.1) WRITE(6,889) ((K,I,J,CM(K,I,J),K=1,3),I=1,3),J=1,3)
C      889 FORMAT(2X,'K=',I5,5X,'I=',I5,5X,'J=',I5,5X,'CM=',IPE12.4)
C
C      DO 8050 K=1,3
C      KK=ELNOD(L,K)
C      DO 8040 I=1,3
C      II=ELNOD(L,I)

```



```

DO 8035 M=1,N*MPK
KKM=NAME(KK,M)
IF(KKM.EQ.0) GO TO 8032
IF(KKM.EQ.1) GO TO 8032
8035 CONTINUE
8032 DO 8030 J=1,3
IF(I.EQ.J) KMN=(M-1)*(2*N*CM*PK-M)/2*PM
IF(I.EQ.J) SCRAPC(KK,KMN)=SCRAPC(KK,KMV)+CM(K,I,J)
IF(I.EQ.J) GO TO 8030
J=ELND(L,J)
DO 8020 N=1,N*MPK
KKN=NAME(KK,N)
IF(KKN.EQ.0) GO TO 8015
IF(KKN.EQ.1) GO TO 8015
8020 CONTINUE
8015 MX=MAX0(M,N)
MN=MIN0(M,N)
KMN=(MN-1)*(N*CM*PK*2-MN)/2+MX
SCRAPC(KK,KMN)=SCRAPC(KK,KMN)+CM(K,I,J)
IF(I.EQ.J) SCRAPC(KK,KMN)=SCRAPC(KK,KMV)+CM(K,I,J)
8030 CONTINUE
8040 CONTINUE
8050 CONTINUE
200 CONTINUE
DO 8050 I=1,NUMSNP
DO 8060 J=1,N*CM
SCRAPC(I,J)=+ZOMEGA*SCRAPC(I,J)*(1.E+14)
8060 CONTINUE
8074 WRITE(6,8074)
FORMAT(1,SCRAPC FROM PSISQ)
DO 8080 K=1,NUMSNP
WRITE(6,8075) (SCRAPC(K,L),L=1,NCC)
WRITE(7,8075) (SCRAPC(K,L),L=1,NCC)
8080 CONTINUE
8075 FORMAT(1P7E15.4)
8076 FORMAT(1P7E11.4)
C
RETURN
C
END

```



```

SUBROUTINE LDASUB(Y,YL,I,TEND,NY,NL,M1,JSTART,KFLAG,MAXOR,IPRI,
1 HMMIV,HMAX, RMSEPS,SAVE,YLSV,YMAX,ER,ESV,FI,DY)
300
400 DIMENSION Y(7,1),YL(1),SAVE(7,1),YMAX(1),ER(1),YLSV(1),FI(1),A(7),
1 PERT(6,3),COF(21),ESV(1),DY(1)
600
700 DATA PERT/4.,9.,16.,25.,36.,49.,9.,16.,25.,36.,49.,64.,1.,1.,
1 .25,2.7889E-2,1.70569E-3,6.83929E-5/
800
900 DATA KZILCH/0/
1000
1100 DATA COF/-1.,-1.5,-.5,-1.833333,-1.,-.1666667,-2.083333,
1200 -1.458333,-.4166667,-.0416667,-2.833333,-1.875,-.7083333,
1 .125,-.08333333,-.2645,-2.255556,-1.208333,-.2430556,
3 -.02916667,-.00138889/
1 FORMAT(215,12,1P2E10.2,7E14.6/(32X,7E14.6))
2 FORMAT(32X,1P2E14.6)
3 FORMAT(
1 TEND =,D9.2,,1 H =,I3,, NL =,I3,, RMSEPS =,1PD9.2,
4 FORMAT(,NS,NW,Q H,8X,T,8X,Y(1,*) AND YL(*))//)
1 IF(JSTART.GT.0)GO TO 60
N = NY + NL
LCOPLY = 7*NY
LCOPLY = NL
EPS = SQRT(FLOAT(M1))*RMSEPS
MAXDER = MAXOR
1 IF(MAXDER.GT.5)MAXDER = 6
1 IF(IPRI.LE.0)GO TO 10
PRINT 3, N,NL,RMSEPS,TEND,H
PRINT 4
10 CONTINUE
NS = 0
NW = 0
DO 20 J=1,NY
YMAX(J) = AMAX1(1.,ABS(Y(1,J)))
Y(2,J) = Y(2,J)*H
NQ = 1
BR = 1.
ASSIGN 100 TO IRET
170 K = NQ*(NQ - 1)/2
CALL COPY2(A(2),COF(K+1),NQ)
230 K = NQ + 1
DOUB = NQ
ENQ1 = NQ
ENQ2 = .5/NQ
ENQ3 = .5/(NQ + 2)
PEPSH = EPS**2
E = PERT(NQ,1)*PEPSH
EUP = PERT(NQ,2)*PEPSH
EDWN = PERT(NQ,3)*PEPSH
BND = (EPS*ENQ3)**2
I MEVAL = 1

```



```

60 GO TO IRET,(100,250,630,751)
   IF(H.EQ.HNEW)GO TO 100
   K = H/HNEW
   ASSIGN 100 TO IRET
   GO TO 800
70 KFLAG = -2
80 JSTART = NQ
   HNEW = H
   RETURN
90 NS = NS + 1
   IF(IPRT.LE.0)GO TO 95
   PRINT 1,NS,NW,NQ,H,T,{Y(1,1),I=1,NY}
   IF(NL.GT.0)PRINT 2,(YL(I),I=1,NL)
95 CONTINUE
   IF(KFLAG.LT.0)GO TO 80
   IF(T.GE.TEND)GO TO 80
   JSTART = 1
100 CALL COPYZ(SAVE,Y,LCPYY)
   CALL COPYZ(YLSV,YL,LCPYL)
   RACUM = 1.
   KFLAG = 1
   HOLD = H
   NQOLD = NQ
   TOLD = T
   KZILCH = 1
   I = T + H
250 HINV = 1./H
   DO 260 J=2,K
   J3 = K+J-1
   DO
   J2 = J3 - J1
   DO 260 I=1,NY
   DO Y(J2,I) = Y(J2,I) + Y(J2+1,I)
260 DO 270 I=1,NY
   ER(I) = 0.
270 CALL NXSTP(Y,YL,T,HINV,A(2),EPS,NY,NL,DY,F1,ER,YMAX,BND,BR,IWEVAL,
1 KFLAG,NW,KRET)
   GO TO {490,440,780},KRET
440 T = TOLD
   IF(IWEVAL)445,455,450
445 IF(H.LE.HMIN*1.00001)GO TO 460
450 RACUM = RACUM*.25
455 CONTINUE
   GO TO 750
460 KFLAG = -3
470 CALL COPYZ(Y,SAVE,LCPYY)
   CALL COPYZ(YL,YLSV,LCPYL)
   H = HOLD

```



```

      NQ = NQOLD
      GO TO 90
490 D = 0
      DO 500 I=1,M1
      YM = AMAX1(ABS(Y(1,I)), YMAX(1))
      D = D + (ER(1)/YM)**2
      IWEVAL = 0
      IF(D.GT.E) GO TO 540
      IF(K.LT.3) GO TO 520
      DO 510 J=3,K
      DO 510 I=1,NY
      Y(J,I) = Y(J,I) + A(J)*ER(I)
510 KFLAG = 1
520 IDOUB = IDOUB - 1
      IF(IDOUB) 550, 525, 700
525 CALL COPYZ(ESV,ER,M1)
      GO TO 700
540 IF(JSTART.GT.3) GO TO 548
      DO 544 I=1,NY
      SAVE(2,I) = Y(2,I)
544 KFLAG = KFLAG - 2 TO 740
548 IF(H.LE.HMIN) GO TO 740
      T = TOLD
      IF(KFLAG.LE.-5) GO TO 720
550 PR2 = (D/E)**ENQ2*1.2
      IF(NQ.LE.1) GO TO 570
      D = 0
      DO 560 J=1,M1
      YM = AMAX1(ABS(Y(1,J)), YMAX(J))
      D = D + (Y(K,J)/YM)**2
560 PR1 = (D/EDWN)**ENQ1*1.3
      IF(PR1.GE.PR2) GO TO 570
      PR2 = PR1
      L = 1
570 IF(KFLAG.LT.0 .OR. NQ.GE.MAXDER) GO TO 590
      D = 0
      DO 580 J=1,M1
      YM = AMAX1(ABS(Y(1,J)), YMAX(J))
      D = D + ((ER(J) - ESV(J))/YM)**2
580 PR1 = (D/EUP)**ENQ3*1.4
      IF(PR1.GE.PR2) GO TO 590
      PR2 = PR1
      L = 1
590 R = 1./AMAX1(PR2,1.E-5)
      IF(KFLAG.LT.0 .OR. R.GE.1.1) GO TO 600
      IDOUB = 9
      GO TO 700

```



```

600 NEWQ = NQ + L
    K = NEWQ + 1
    IF (NEWQ.LE.NQ) GO TO 620
    R1 = A(NEWQ)/FLOAT(NEWQ)
    DO 610 J=1,NY
        Y(K,J) = ER(J)*R1
610 CONTINUE
    IDOUB = NQ
    IF (NEWQ.EQ.NQ) GO TO 630
    NQ = NEWQ
    ASSIGN 630 TO IRET
    GO TO 170
630 IF (KFLAG.GT.0) GO TO 670
    RACUM = RACUM*R
    GO TO 750
670 R = AMAX1(AMINI(HMAX/H,R),HMIN/H)
    IWEVAL = 1
    ASSIGN 700 TO IRET
    GO TO 800
700 DO 710 I=1,M1
710 YMAX(I) = AMAX1(ABS(Y(I,I)),YMAX(I))
720 IF (NQ.EQ.1) GO TO 735
    NQ = 1
    IDOUB = 1
    ASSIGN 751 TO IRET
    GO TO 170
735 NGOLD = 1
    KFLAG = -4
    GO TO 470
740 KFLAG = -1
    GO TO 90
750 H = HOLD*RACUM
751 RACUM = H/HOLD
    R1 = 1.
    DO 760 J=2,K
        R1 = R1*RACUM
        DO 760 I=1,NY
            Y(J,I) = SAVE(J,I)*R1
760 DO 770 I=1,NY
770 Y(I,I) = SAVE(I,I)
    CALL CJPV2(YL,YLSV,LCOPYL)
    IWEVAL = 1
    GO TO 250
780 KFLAG = -5
    GO TO 80

```

17200
17900
17400
17500
17600
17700
17800
17900
18000
18100
18200
18300
18400
18500
18600
18700
18800
18900
19000
19100
19200
19300
19400
19500
19600
19700
19800
19900
20000
20100
20200
20300
20400
20500
20600
20700
20800
20900
21000
21100
21200
21300
21400
21500
21600
21700
21710
21720


```

800 R1 = 1. J=2,K
DO 810 R1 = R1*R
DO 810 I=1,NY
Y(J,I) = Y(J,I)*R1
GO TO IRET,(100,700)
ENTRY REDSUB
IF(KZILCH.EQ.0)RETURN
DO 910 I=1,NY
Y(1,I) = SAVE(1,I)
Y(2,I) = SAVE(2,I)/HOLD
CALL COPYZ(YL,YLSV,LCOPYL)
T = TOLD
RETURN
END

```

```

21800
21900
22000
22100
22200
22300
22400
22500
22600
22700
22800
22900
23000
23100
23200

```



```

SUBROUTINE NXSTP(Y, YL, T, HINV, A2, EPS, NY, NL, DY, FL, ER, YMAX, BND, BR,
1 IWEVAL, KFLAG, NW, KRET)
DIMENSION Y(7,1), YL(1), DY(1), ER(1), F1(1), YMAX(1), PW(110,7)
COMMON/CMCL/A(132,7), B(132,7), C(132,28), N, NNZ, K(132,7)
INTEGER*2 K
DATA SPD, SPDM1/1.05,.05/
KRET = 1
DO 430 L=1,3
CALL DIFFUN(Y, YL, T, HINV, DY)
CALL IF(IWEVAL, LT, 1) GO TO 280
NOIT = N**2
EPSS = EPS**2
EPSA2 = EPSS**2001
CALL JACMAT(Y, YL, T, HINV, A2, EPS, NY, NL, DY, FL, PW)
KFLAG = 1
IWEVAL = -1
NW = NW + 1
280 DO 281 I=1, NY
281 F1(I) = DY(I)/PW(I,1)
DO 287 IT = 1, NOIT
RCH = 0.
CH = 0.
DO 285 I=1, NY
FN = DY(I)
DO 284 J=2, NNZ
IF(K(I, J).LE.0. OR K(I, J).GT. NY) GO TO 284
FN = FN - PW(I, J)*F1(K(I, J))
CONTINUE
FN = FN/PW(I,1)
FN = FN*SPD - SPDM1*F1(I)
ACH = F1(I) - FN
ACH = CH + (ACH/YMAX(I))**2
RCH = RCH + (ACH/AMAX1(ABS(FN), EPS))**2
285 F1(I) = FN
IF(RCH.LT.EPSS) GO TO 288
IF(CH.LE.EPSA2) GO TO 288
287 CONTINUE
KRET = 3
RETURN
288 CONTINUE
IF(NL.LE.0) GO TO 300
DO 290 I=1, NL
290 YL(I) = YL(I) - F1(I+NY)
300 CONTINUE
DEL=0.
DO 420 I=1, NY
Y(1, I) = Y(1, I) - F1(I)
Y(2, I) = Y(2, I) + A2*F1(I)

```



```

      ER(I) = ER(I) + F1(I)
      DEL = DEL + (F1(I)/AMAX1(YMAX(I),ABS(Y(1,I))))**2
      CONTINUE
      IF(LGE.2)BR = AMAX1(.9*BR,DEL/DEL1)
      DEL1 = DEL
      IF(AMINI(DEL,BR*DEL*2.).LE.BND)GO TO 450
      CONTINUE
      KRET = 2
      RETURN
      CONTINUE
      RETURN
      END
420
430
450

```



```

SUBROUTINE SDESOL(Y,YL,T,TEND, NY,NL,M,JSKF,MAXDER, IPRT, H,
1 HMIN,HMAX,EPS,W)
  DIMENSION Y(7,1),YL(1),W(1)
  IF(JSKF.NE.0) GO TO 200
  DO 110 I=1,NY
    Y(2,I) = 0.
  110 CALL DIFFUN(Y,YL,0.,1.,W)
  CALL DERRVAL(W,KRETR)
  IF(KRETR.NE.0) GO TO 300
  DO 120 I=1,NY
    Y(2,I) = W(1)
  120 N = NY + NL
    NSV = 1
    NSVL = 7*NY + 1
    NYMAX = NSVL + NL
    NER = NYMAX + 1
    NESV = NER + NY
    NF1 = NESV + NY
    NDY = NF1 + N
  200 JS = JSKF
    CALL LDASUB(Y,YL,T,TEND, NY,NL,M,JS,KF,MAXDER, IPRT, H,HMIN,
1 HMAX,EPS,W,NSVL),W(NYMAX),W(NSVL),W(NER),W(NESV),W(NF1),W(NDY))
    JSKF = ISIGN(JS*10 + IABS(KF),KF)
  RETURN -6
  300 JSKF = -6
    RETURN
  END

```



```

SUBROUTINE Derval(DY,KERET)
COMMON/CMCL/A(132,7),B(132,7),C(132,28),N,NNZ,K(132,7)
INTEGER*2 K
DIMENSION DY(1)
DO 100 I=1,N
  DY(I+N) = -DY(I)
100  DY(I) = -DY(I)/A(I,1)
DO 200 L=1,500
DO 180 I=1,N
  DYN = DY(I+N)
  E = 0
DO 150 J=2,NNZ
  IF(K(I,J).LE.0,OR-K(I,J).GT-N)GO TO 150
  DYN = DYN - A(I,J)*DY(K(I,J))
150  CONTINUE
  DYN = DYN/A(I,1)
  E = AMAX1(ABS{(DYN - DY(I))/DYN},E)
180  DY(I) = DYN
  IF(E.LT.1.E-4)GO TO 300
200  CONTINUE
  PRINT 1,E
  IF(E.LE.1.E-2)GO TO 300
  KERET = 1
  RETURN
300  KERET = 0
  RETURN
  1  FORMAT(' AFTER 500 ITERATIONS, E =',1PE10.3)
END

```



```

SUBROUTINE JACMAT(Y,YL,T,HINV,A2,EPS,NY,NL,DY,F1,PW)
COMMON/CMCL/A(132,7),B(132,7),C(132,28),N,NNZ,K(132,7)
INTEGER*2 K,P
DIMENSION Y(7,1),YL(1),F1(1 ),DY(1),PW(110,1)
DIMENSION P(8,8)
DATA P/64*0/,INITP/0/
IF(INITP.EQ.NNZ)GO TO 99
INITP = NNZ
DO 98 L=1,NNZ
DO 98 M = L,NNZ
P(L,M) = M + (L - 1)*(2*NNZ - L)/2
98 CONTINUE
AH =-A2*HINV
DO 300 I=1,NY
DO 300 J=1,NNZ
PW(I,J) = AH*A(I,J) - B(I,J)
DO 100 L=1,J
IF(K(I,L).LE.0)GO TO 295
PW(I,J) = PW(I,J) + C(I,P(L,J))*Y(L,K(I,L))
100 DO 200 M=J,NNZ
IF(K(I,M).LE.0)GO TO 295
PW(I,J) = PW(I,J) + C(I,P(J,M))*Y(L,K(I,M))
200 CONTINUE
295 CONTINUE
300 RETURN
END

```



```

SUBROUTINE COPYZ(S,Y,L)
  DIMENSION S(1),Y(1)
  IF(L.LE.0)RETURN
  DO 100 J=1,L
    S(J)=Y(J)
  RETURN
END

100

SUBROUTINE DIFFUN(Y,YL,I,HINV,DY)
  COMMON/CMCL/A(132,7),B(132,7),C(132,28),N,NNZ,K(132,7)
  INTEGER*2 K
  DIMENSION Y(7,1),YL(1),DY(1)
  DO 400 I=1,N
    DY(I)=0.
    DO 300 J=1,NNZ
      IF(K(I,J).LE.0)GO TO 310
      DY(I)=DY(I)+Y(2,K(I,J))*A(I,J)*HINV - B(I,J)*Y(1,K(I,J)) +
1      C(I,J)*Y(1,K(I,J))*Y(1,K(I,1))
300 CONTINUE
310 L=NNZ
    DO 360 J1=2,NNZ
      IF(K(I,J1).LE.0)GO TO 400
      DO 350 J2=J1,NNZ
        L=L+1
        IF(K(I,J2).LE.0)GO TO 350
        DY(I)=DY(I)+C(I,L)*Y(1,K(I,J1))*Y(1,K(I,J2))
350 CONTINUE
360 CONTINUE
400 RETURN
END

```



```

*****
**
**      PROGRAM FED-2-COMPACT
**
**      FINITE ELEMENT SOLUTION OF NON-LINEAR REACTOR
**      DYNAMICS IN TWO DIMENSIONAL SPACE WITH PRE-
**      MULTIPLICATION TO REDUCE THE NON-LINEARITY.
**
*****
**
**      SOLUTION METHOD      1-CRANKO (INCLUDED)
**
**      1- CRANKO ROUTINE BASED ON CRANK-NICOLSON
**      FORMULATION AND SOLVED BY GAUSS-SEIDEL
**      ITERATION.
**
**      INPUT PARAMETERS AND DIMENSIONING REQUIREMENTS.
**
**      SINGLE ITEMS
**      NUMSNP IS THE NUMBER OF SYSTEM NODAL POINTS.
**      NUMEL IS THE NUMBER OF TRIANGULAR ELEMENTS.
**      NUPBP IS THE NUMBER OF EXTERIOR BOUNDARY PTS.
**      NUFUEL IS THE NUMBER OF ELEMENTS IN THE CORE.
**      ERRVAL IS THE CONVERGENCE CRITERIA FOR CRANKO
**      TO IS THE PROBLEM START TIME
**      TF IS THE PROBLEM FINISH TIME
**      HMIN IS THE FIRST ATTEMPTED TIME INTERVAL
**      HMAX IS THE SMALLEST TIME INTERVAL
**      LCON IS THE LARGEST ALLOWED TIME INTERVAL
**      LCON IS THE MAX NUMBER OF NODE CONTRIBUTORS
**
**      THESE ARE THE INPUT REQUIREMENTS
**      TITLE VJREL NUPBP NUMSNP NEUEL IUPBP ERRVAL
**      AFUEL TO H TF HMIN HMAX V D SGA SGF ALPHA
**      PSIIV SYSNOD R Z ELEMNT ELNOD ITYPE LCON MNOD
**
**      ITYPE=0 DENOTES CORE ELEMENTS
**      ITYPE=1 DENOTES BLANKET ELEMENTS
**
*****

```



```

THESE ARE THE DIMENSIONING REQUIREMENTS.
PART(NUMSNP,LCON),BIGA(NUMSNP,LCON),BIGC(NUMSNP,LCON),BIGCC(NUMSNP,LCON),
BIGAB(NUMSNP,LCON),BIGC(NUMSNP,LCON),D(NUMEL),SGA(NUMEL),
MNODE(NUMSNP,MNODE),V(NUMEL),D(NUMEL),SGA(NUMEL),
SGF(NUMEL),ALPHA(NUMEL),PSI1V(NUMSNP),PDP1(NUMSNP),
ZLMBDA(NUMEL),VLMBDA(NUMEL),ZOMEGA(NUMEL),
ELEMNT(NUMEL),SYSNOD(NUMSNP),R(NUMSNP),Z(NUMSNP),
Z1(NUMEL),R2(NUMEL),Z2(NUMEL),R3(NUMEL),Z3(NUMEL),
A(NUMEL,3),B(NUMEL,3)
ERROR(NUMSNP),CM(NUMEL,3,3),ALFA(3,3),PSTEP(NUMSNP),
BIGD(NUMSNP,LCON),BIGE(NUMSNP,LCON),ESTR(NUMSNP),VD(NUMEL)
YMAX(NUMSNP),IUPBP(NUMBP),PSI(8,NUMSNP),DP,SI(NUMSNP),
AREA(NUMEL),PV(NUMSNP),HOLD(NUMSNP)

```

----- DECLARATION STATEMENTS -----

```

NONLINEAR SUPERCRITICAL REACTOR
PROMPT FEEDBACK
FULLY REFLECTED SYSTEM
SPACE-DEPENDENT NEUTRONIC PROPERTIES

```

*****@


```

C      INTEGER*4 SYSNOD,ELEMNT,ELNOD
C      DIMENSION TITLE(20)

      DIMENSION PART(132,7),MNODE(132,7),BIGA(132,7),BIGC(132,7),
1     BIGCC(132,7),PSIIV(132),POPSI(132),VLMBDA(220),SGA(220),SGF(220),
2     ALPHA(220),RHO(220),SYSNOD(132),R(132),Z(132),VLMBDA(220),
3     ZOMEGA(220),ITYPE(220),R1(220),R2(220),Z1(220),Z2(220),
4     ELVJOD(220,3),A(220,3),P(224,4),BAT(400,4),ERROR(132),
5     R3(220,3),CM(220,3,3),PSTEP(132),BIGD(132,7),BIGE(132,7),ESTR(132),
6     CM(220,3,3),YMAX(132),IUPBP(22),PSI(8,132),AREA(220),PV(132),
7     VD(220),YMAX(132),IUPBP(22),PSI(8,132),AREA(220),PV(132),
8     B(220,5),ASTR(132,7),HOLD(132),WP(132,132)

C      READ(5,998) TITLE
C      READ(5,10) NUMEL,NUPBP,NUMSNP,NFULEL
998    FORMAT(20A4)
10    FORMAT(8I10)

C      LCON=7
C      CALL ERRSET(207,256,0,1,1,209)

C      *
C      CALL FEDMAN (NUMEL,NUPBP,NUMSNP,NFULEL,LCON,TITLE,
1     ALPHA,PSIIV,POPSI,ZLMBDA,VLMBDA,ZOMEGA,
2     ELNOD,ITYPE,R1,Z1,R2,Z2,R3,Z3,A,B,
3     PSTEP,BIGD,BIGE,ASTR,ESTR,VD,YMAX,IJPPBP,AREA,PV,HOLD)
4     *
C      * STOP
C      END

```



```

SUBROUTINE FEMAN (NJMEL,NJPBP,NUMSNP,VFULEL,LCON,TITLE,
1  ALPHA,PSIIV,PDPSI,BIGAB,BIGC,BIGCC,MNOD,V,D,SGA,SGF,
2  ELNOD,ITYPE,R1,Z1,R2,Z2,R3,Z3,A,B,SYSNOD,R,Z,ELEMNT,
3  PSTEP,BIGD,BIGE,ASTR,ESTR,R,VD,YMAX,IJPBP,AREA,PV,HOLD)
4  * * * * *
      FINITE ELEMENT SOLUTION OF NONLINEAR
      REACTOR DYNAMICS IN TWO-DIMENSIONAL SPACE
      * * * * *
      INTEGER*4 SYSVJD,ELEMNT,ELNOD
      DIMENSION TITLE(20)
      DIMENSION PART(NUMSNP,LCON),BIGA(NUMSNP,LCON),
1  BIGAB(NUMSNP,LCON),BIGC(NJMSNP,LCON),BIGCC(NUMSNP,LCON),
2  MNOD(NUMSNP,LCON),V(NUMEL),D(NUMEL),SGA(NUMEL),
3  SGF(NUMEL),ALPHA(NUMEL),PSIIV(NUMSNP),PDPSI(NUMSNP)
      DIMENSION ZLMBDA(NUMEL),VLMBDA(NUMEL),ZOMEGA(NUMEL),
1  SYSNOD(NUMSNP),R(NUMSNP),Z(NUMSNP)
2  ELEMNT(NUMEL),ELNOD(NUMEL,3),R3(NJMEL),Z3(NUMEL),
3  Z1(NUMEL),R2(NUMEL),Z2(NUMEL),R3(NJMEL),Z3(NUMEL)
      DIMENSION A(NUMEL,3),B(NUMEL,3),
1  ERROR(NUMSNP),CM(NUMEL,3,3,3),ALFA(3,3),PSTEP(NUMSNP),
2  BIGD(NUMSNP,LCON),BIGE(NUMSNP,LCON),ESTR(NUMSNP),VD(NUMEL),
3  YMAX(NUMSNP),IJPBP(NJPBP),AREA(NUMEL),PV(NUMSNP),HOLD(NUMSNP)
      DO 05 I=1,NUMSNP
        ERROR(I)=0.
        PDPSI(I)=0.
        YMAX(I)=1.
05  CONTINUE
      NUF=NUMSNP/5
      XNP=NUMSNP
      XNUF=XNP/5.
      IF ((XNUF-FLOAT(NUF)) .LT. 1.) NUF=NUF+1
      * * * * *
      GENERAL PROBLEM DATA
      READ(5,12) (IJPBP(I),I=1,NJPBP)
      READ(5,10) MTH,MAXDER,NCOUNT
10  FORMAT(8I10)
12  FORMAT(16I5)

```



```

C
      WRITE(6,11) NUMEL,NUPBP,NUMSNP,NFUELE
11  FORMAT(/,2X,'# OF ELEMENTS=',I5/,2X,'# OF UPPER BDRY NODAL POINTS=',
1,15/,2X,'# OF SYSTEM NODAL POINTS=',I5/,2X,'# OF FUEL ELEMENTS=',
215)
      READ(5,15) ZNJ,FISFAC,HBAR,EPSVAL,ERRVAL,AFUEL
      READ(5,19) TO,H1,F,HMIN,HMAX
      READ(5,15) (V(I),I=1,NUMEL)
      READ(5,15) (D(I),I=1,NUMEL)
      READ(5,15) (SGA(I),I=1,NUMEL)
      READ(5,15) (SGF(I),I=1,NUMEL)
      READ(5,1000) ALPHA(I),I=1,NUMEL)
      READ(5,18) (PSIV(I),I=1,NUMSNP)
15  FORMAT(8G10.5)
1000 FORMAT(5F15.8)
C
C
C
      CALCULATE PHYSICAL CONSTANTS FROM NUCLEAR DATA
      PI= 3.1415927
      DO 25 I=1,NUMEL
      VD(I)=V(I)*D(I)
      ZLMBDA(I)=ZNU*SGF(I)/SGA(I)-1.
      VLMBDA(I)=V(I)*ZLMBDA(I)*SGA(I)
      ZOMEGA(I)=V(I)*SGA(I)*ALPHA(I)*FISFAC*SGF(I)/HBAR
25  CONTINUE
C
      18  FORMAT(5E15.4)
      19  FORMAT(2E20.4,4G10.5)
      998 FORMAT(20A4)
C
C
C----- GEOMETRY OF SYSTEM NODAL POINTS -----
C
      WRITE(6,40)
40  FORMAT(/,1X,'GEOMETRY OF SYSTEM NODAL POINTS',/)
C
      READ(5,41) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)
41  FORMAT(5X,15,2F15.5)
C
      WRITE(6,60) (SYSNOD(I),R(I),Z(I),I=1,NUMSNP)
60  FORMAT(1X,'SYSTEM NODAL POINT NO.',13,5X,'R COORD.= ',F10.6,5X,'
1Z COORD.= ',F10.6)
C
C
C

```



```

C      WRITE(6,95)
95      FORMAT(1X,///1X,'ELEMENT',5X,'R1',9X,'Z1',9X,'R2',9X,'Z2',9X,'R3',
C      19X,'Z3,/')
C      DO 100 I=1,NUMEL
C      J=ELNOD(I,1)
C      R1(I)=R(J)
C      Z1(I)=Z(J)
C      K=ELNOD(I,2)
C      R2(I)=R(K)
C      Z2(I)=Z(K)
C      L=ELNOD(I,3)
C      R3(I)=R(L)
C      Z3(I)=Z(L)
C      WRITE      (6,105) I,R1(I),Z1(I),R2(I),Z2(I),R3(I),Z3(I)
C      100 CONTINUE
105      FORMAT (3X,I3,3X,      7(F10.6,1X))
C      COMPUTE A1,A2,A3,B1,B2,B3,AREA FOR EACH ELEMENT
C      -----
C      120      WRITE      (6,120)
120      FORMAT(1X,///1X,'ELEMENT',5X,'A1',11X,'A2',11X,'A3',11X,'B1',11X,
C      1'B2',11X,'B3',10X,'AREA,/')
C      DO 140 I=1,NUMEL
C      A(I,1)=R3(I)-R2(I)
C      A(I,2)=R1(I)-R3(I)
C      A(I,3)=R2(I)-R1(I)
C      B(I,1)=Z2(I)-Z3(I)
C      B(I,2)=Z3(I)-Z1(I)
C      B(I,3)=Z1(I)-Z2(I)
C      AREA(I)=0.5*(A(I,2)*B(I,1)-A(I,1)*B(I,2))
C      WRITE(6,130) I,A(I,1),A(I,2),A(I,3),B(I,1),B(I,2),B(I,3),
C      1'AREA(I)
140      CONTINUE
C      130      FORMAT (3X,I3,3X,      7(F12.7,1X))
C
C

```



```

C----- ZERO THE BIGA,      BIGC AND BIGAB MATRICES-----
CCCC
DO 194 KK=1,NUMSNP
DO 193 II=1,LCON
BIGA(KK,II)=0.0
BIGAB(KK,II)=0.0
BIGC(KK,II)=0.0
PART(KK,II)=0.0
193 CONTINUE
194 CONTINUE
*****
C-----
C----- PEPSI CALCULATES THE BIGA MATRIX AND PART OF THE BIGAB
C-----
CALL PEPSI (NJVEL,NUMSNP,R1,R2,R3,AREA,ELNOD,LCON,MNOD,
1 BIGA,BIGAB,VLMBOA)
C
195 FORMAT(/5X,'BIGA MATRIX',/)
WRITE(6,195)
778 FORMAT(2X,7(E14.5))
WRITE(6,196)
196 FORMAT(/5X,'BIGAB MATRIX FROM PEPSI, NO DEL**2 CONTRIBUTION',/)
WRITE(6,778) ((BIGAB(I,J),J=1,LCON),I=1,NUMSNP)
C-----
C----- CRUISE CALCULATES THE BIGAB MATRIX
C-----
CALL CRUISE (NJMEL,NUMSNP,LCON,R1,R2,R3,AREA,ELNOD,
1 BIGAB,MNOD,A,B,VD)
C
197 FORMAT(/5X,'3IGAB MATRIX FROM CRUISE, WITH DEL**2 TERM',/)
WRITE(6,778) ((BIGAB(I,J),J=1,LCON),I=1,NUMSNP)
C-----
C-----
DO 255 I=1,NUPBP
I1=IUPBP(I)

```



```
C
C      200 CONTINUE
C      151 CONTINUE
C      C
C      IF (M1 .EQ. 5) CALL CRANKJ (NUMEL, NUMSNP, NUPBP, LCON, H, TO,
1     ERVAL, EPSVAL, IUPBP, PSIV, ITYPE, ELVDD, CM, MNOD, ZOMEGA,
2     BIGAB, BIGA, PV, PSTEP, BIGD, BIGE, ASTR, ESTR,
3     BIGC, BIGCC, HOLD, TF)
C
C      STOP
C      END
```



```

SUBROUTINE PEPSI (NUMEL,NUMSNP,R1,R2,R3,AREA,ELNOD,LCON,MNOD,
1 BIGA,BIGAB,VLMBDA)
C *****
C PEPSI CALCULATES THE BIGA MATRIX AND PART OF THE BIGAB
C *****
C INTEGER*4 ELNOD
C
C DIMENSION R1(NJMEL),R2(NUMEL),R3(NJMEL),BIGA(NUMSNP,LCON),
1 BIGAB(NUMSNP,LCON),MNOD(NUMSNP,LCON),AMATRIX(3,3),
2 VLMBDA(NUMEL),AREA(NUMEL),ELNOD(NUMEL,3)
C
C PI=3.1415927
C
C CALCULATE THE 3X3 D(I,J) MATRIX FOR ELEMENT L
C
DO 200 L=1,NUMEL
C   COEFFA=(PI/30.0)*AREA(L)
C   AMATRIX(1,1)=COEFFA *(6.0*R1(L)+2.0*R2(L)+2.0*R3(L))
C   AMATRIX(1,2)=COEFFA *(2.0*R1(L)+2.0*R2(L)+2.0*R3(L))
C   AMATRIX(1,3)=COEFFA *(2.0*R1(L)+R2(L)+2.0*R3(L))
C   AMATRIX(2,1)=AMATRIX(1,2)
C   AMATRIX(2,2)=COEFFA *(2.0*R1(L)+6.0*R2(L)+2.0*R3(L))
C   AMATRIX(2,3)=COEFFA *(R1(L)+2.0*R2(L)+2.0*R3(L))
C   AMATRIX(3,1)=AMATRIX(1,3)
C   AMATRIX(3,2)=AMATRIX(2,3)
C   AMATRIX(3,3)=COEFFA *(2.0*R1(L)+2.0*R2(L)+6.0*R3(L))
C
C-----
C STORE ELEMENT MATRIX, AMATRIX, INTO SYSTEM MATRIX, BIGA
C-----
C
DO 20 K=1,3
KK=ELNOD(L,K)
DO 10 I=1,3
II=ELNOD(L,I)
DO 91 MX=1,LCON
NOW =MX
MM =MNOD(KK,MX)
IF (MM .EQ. II) GO TO 92
91 CONTINUE
92 CONTINUE
BIGA(KK,NOW)=BIGA(KK,NOW)+AMATRIX(K,I)*AMATRIX(K,I)
10 CONTINUE
20 CONTINUE
200 RETURN
END

```


SUBROUTINE CRJISE (NUMEL, NUMSNP, LCON, R1, R2, R3, AREA, ELNOD, BIGAB,
 MNOD, A, B, VJ) *

CRUISE CALCULATES THE BIGAB MATRIX

INTEGER*4 ELNOD

DIMENSION R1(NUMEL), R2(NUMEL), R3(NUMEL), AREA(NUMEL),
 1 MNOD(NUMSNP, LCON), BIGAB(NUMSNP, LCON), BMATRIX(3, 3),
 2 ELNOD(NUMEL, 3), A(NUMEL, 3), B(NUMEL, 3), VD(NUMEL)

PI=3.1415927

CALCULATE THE 3X3 B(I,J) MATRIX FOR ELEMENT L

DO 200 L=1, NUMEL

COEFFB=PI*VD(L)/(6.0*AREA(L))

BMATRIX(1,1)=COEFFB *((-R1(L))*(2.0*B(L,1)**2+2.0*A(L,1)**2)-R2(L)*
 1 B(L,1)*B(L,2))+B(L,1)**2+A(L,1)**2)-R3(L)**2*B(L,1)*
 2 B(L,3))+B(L,1)**2+A(L,1)**2)+2.0*AREA(L)*B(L,1)**

BMATRIX(1,2)=COEFFB *((-R1(L))*(2.0*B(L,1)*B(L,2)+2.0*A(L,1)*
 1 A(L,2)*B(L,2))+B(L,2)**2+B(L,1)*B(L,2)+A(L,2)**2+A(L,1)*A(L,2))-
 2 R3(L)*B(L,2)*B(L,3))+B(L,1)*B(L,2)+A(L,2)*A(L,3)+A(L,1)*A(L,2))+
 3 2.0*AREA(L)*B(L,2)**

BMATRIX(1,3)=COEFFB *((-R1(L))*(2.0*B(L,1)*B(L,3)+2.0*A(L,1)*
 1 A(L,3)*B(L,3))+B(L,3)**2+B(L,1)*B(L,3)+A(L,3)**2+A(L,1)*A(L,3))+
 2 A(L,3))-R3(L)*B(L,3)*B(L,3))+B(L,1)*B(L,3)+A(L,1)*A(L,3))+
 3 2.0*AREA(L)*B(L,3)**

BMATRIX(2,1)=BMATRIX(1,2)

BMATRIX(2,2)=CJEFFB *((-R1(L))*(B(L,2)**2+B(L,1)*B(L,2)+A(L,2)**2+
 1 A(L,1)*A(L,2))-R2(L)*(2.0*B(L,2)**2+2.0*A(L,2)**2)-R3(L)*B(L,3)*
 2 B(L,2))+B(L,2)**2+A(L,3)*A(L,2)**2)+2.0*AREA(L)*B(L,2)**

BMATRIX(2,3)=COEFFB *((-R1(L))*(B(L,2)*B(L,3)+B(L,1)*B(L,3)+A(L,2)*
 1 A(L,3)+A(L,1)*A(L,3))-R2(L)*(2.0*B(L,2)*B(L,3)+2.0*A(L,2)*A(L,3))-
 2 R3(L)*B(L,3)*B(L,2))+B(L,2)*B(L,3)+A(L,3)*A(L,2)*A(L,3))+
 3 2.0*AREA(L)*B(L,3)**


```

C      BMATRX(3,1)=BMATRX(1,3)
C      BMATRX(3,2)=BMATRX(2,3)
C      BMATRX(3,3)=CJEEFB *(-R1(L)*(B(L,3)**2+B(L,1)*B(L,3)+A(L,3)**2+
1A(L,1)*A(L,3))-R2(L)*(B(L,3)**2+B(L,2)*B(L,3)+A(L,2)**2+A(L,2)*
2A(L,3))-R3(L)*(2.0*B(L,3)**2+2.0*A(L,3)**2)+2.0*AREA(L)*B(L,3))
C
C      STORE ELEMENT MATRIX, AMATRX, INTO SYSTEM MATRIX, BIGA
C-----
C      DO 20 K=1,3
C      KK=ELNOD(L,K)
C      DO 10 I=1,3
C      II=ELNOD(L,I)
C      DO 91 MX=1,LCON
C      NOW=MX
C      MM=MNOD(KK,MX)
C      IF (MM .EQ. 11) GO TO 92
C      91 CONTINUE
C      92 CONTINUE
C      BIGAB(KK,NOW)=3IGAB(KK,NOW)+BMATRX(K,I)
C      10 CONTINUE
C      20 CONTINUE
C      C 200 CONTINUE
C      C 200 CONTINUE
C      RETURN
C      END

```



```

SUBROUTINE CRANKO (NJMEL,NUMSNP,NUPBP,LCON,H,TO,ERRVAL,
1 EPSVAL,IUP3P,PSIIV,ITYPE,ELNOD,CM,MNOD,ZOMEGA,BIGAB,BIGA,
2 PV,PSTEP,BIGD,BIGE,ASTR,ESTR,
3 BIGC,BIGCC,HOLD,TF)

```

CC
CC
C

```

INTEGER*4 ELNJO

```

```

DIMENSION IUPBP(NUPBP),PSIIV(NUMSNP),ITYPE(NUMEL),
1 ELNOD(NUMEL,3),CM(NUMEL,3,3),MNOD(NUMSNP,LCON),
2 ZOMEGA(NUMEL,3),BIGAB(NUMSNP,LCON),BIGA(NUMSNP,LCON),
3 PSTEP(NUMSNP),ALFA(3,3),PV(NUMSNP)
DIMENSION BIGD(NUMSNP,LCON),BIGE(NUMSNP,LCON),
1 ASTR(NUMSNP,LCON),ESTR(NUMSNP),BIGC(NUMSNP,LCON),
2 BIGCC(NUMSNP,LCON),HOLD(NUMSNP)

```

CC
CC
CC
CC
CC
CC

```

DLT IS THE MAX TIME INTERVAL ATTEMPTED FOR ONE STEP.
EPSN IS THE CONVERGENCE CRITERIA FOR SOLN AT TIME T=T+DT

```

INITIALIZATION

```

DLT=H
PTIME=H
T=TO
LRGE = 1
EPSN=ERRVAL
STAR=EPSVAL
NIT=0
NUMEQ=NUMSNP-NJPBP
MIN=NUMSNP/4

```

CC

```

NUF=NUMSNP/5
XNP=NUMSNP
XNUF=XNP/5
IF ((XNUF-FLOAT(NUF)) .LT. 1.) NUF=NUF+1

```

C

```

DO 05 I=1,NUMSNP
PV(I)=0
05 CONTINUE

```

CC
CC
CC

```

BUILD THE BIGCC MATRIX

```

```

351 CONTINUE

```



```

C
DO 07 I=1,NUPBP
  J=IUPBP(I)
  PSIIV(J)=0.
07 CONTINUE
DO 09 I=1,NUMSNP
  PSTEP(I)=PSIIV(I)
09 CONTINUE

C
DO 210 L=1,NUMEL
  LL=ITYPE(L)
  IF(LL.NE.0) GO TO 210
  N1=ELNOD(L,1)
  N2=ELNOD(L,2)
  N3=ELNOD(L,3)
  DO 88 J=1,3
    DO 90 N=1,3
      ALFA(N,J)=PSIIV(N1)*CM(L,N,J,1)+PSIIV(N2)*CM(L,N,J,2)
      +PSIIV(N3)*CM(L,N,J,3)
1 CONTINUE
90 CONTINUE
88 CONTINUE
DO 150 K=1,3
  KK=ELNOD(L,K)
  DO 180 I=1,3
    II=ELNOD(L,I)
    DO 91 MX=1,LCON
      NOW=MX
      MM=MNOD(KK,MX)
      IF(MM.EQ.II) GO TO 92
91 CONTINUE
92 CONTINUE
  BIGC(KK,NOW)=BIGC(KK,NOW)+ALFA(K,I)*ZDMEGA(L)
180 CONTINUE
150 CONTINUE
210 CONTINUE

C
C
C      INSERTION OF BOUNDARY POINTS
DO 185 I=1,NUPBP
  II=IUPBP(I)
  DO 191 MX=1,LCON
    BIGC(II,MX)=0
191 CONTINUE
185 CONTINUE

C
C
DO 310 I=1,NUMSNP
  DO 310 J=1,LCON
    BIGCC(I,J)=BIGAB(I,J)-BIGC(I,J)

```



```

310 CONTINUE
      BEGIN STEPIING OUT IN TIME.
      STP = 4.
      ITRY=0
      DO 55 JQ=1,12
        COMPUTE PSI BASED ON A TRIAL DT.
        FIRST COMPJTE (2/DT)*A=ASTR, ASTR-C=D, ASTR+D=E
        DS=2./DLI
        DO 15 KA=1,NUMSNP
          DO 13 KB=1,LCON
            ASTR(KA,KB)=DS*3 IGA(KA,KB)
            BIGD(KA,KB)=ASTR(KA,KB)-BIGCC(KA,KB)
            BIGE(KA,KB)=ASTR(KA,KB)+BIGCC(KA,KB)
          13 CONTINUE
        15 CONTINUE
        DO 19 KA=1,NUMSNP
          ESTR(KA)=0.
          DO 17 KB=1,LCON
            NAME = MNOD(KA,KB)
            ESTR(KA)=ESTR(KA)+BIGE(KA,KB)*PSIIV(NAME)
          17 CONTINUE
        19 CONTINUE
        THE SYSTEM IS NOW IS THE FAMILIAR FORM D(I,J)*PV(J)=ESTR(J)
        AND MAY BE SOLVED USING ANY TECHNIQUE FOR LINEAR SIMULTANEOUS
        EQNS. HERE GAUSS-SEIDEL ITERATION IS USED.
        ACCOMPLISH THE FIRST ITERATION BASED ON THE PV VALUE AT THE
        LAST SUCCESSFUL TIME POINT. SECOND AND SUCCESSIVE ITERATIONS
        ARE BASED ON UPDATED VALUED OF PV. A TEST FOR CONVERGENCE IS
        MADE, AND FAILURE RESULTS IN ANOTHER ATTEMPT WITH A SMALLER DT
        CREATION OF ESTR(I)=BIGE(I,J)*PSIIV(I)
        PERFORMED ONCE ON EACH ATTEMPT TO FIND A PROPER TIME INTERVAL.
        DO 45 MTRY=1,NUMEQ
          KT=0
          DO 25 K=1,NUMEQ
            HOLD(K)=0.
            PV(K)=0.
          25 CONTINUE
          DO 20 L=2,LCON
            NAME = MNOD(K,L)
            ESTR(K)=ESTR(K)+BIGE(K,L)*PSIIV(L)
          20 CONTINUE
          IF (ABS(ESTR(K)-ESTR(K-1))) .GT. 1E-6) GO TO 45
          KT=KT+1
          IF (KT .EQ. 10) GO TO 45
        45 CONTINUE
      55 CONTINUE

```



```

HOLD(K)= HOLD(K) + BIGD(K,L)*PSTEP(NAME)
20 CONTINUE
PV(K)=(ESTR(K)-HOLD(K))/BIGD(K,1)
PDIF=(PV(K)-PSTEP(K))/PV(K)
24 CONTINUE
IF (ABS(PDIF) .LT. EPSN) KT =KT+1
PSTEP(K)=PV(K)
25 CONTINUE
IF (KT .GE. NJMEQ) GO TO 65
45 CONTINUE

C
C 291 FORMAT (//10X,'NO SOLN FOUND IN INTERVAL ',G10.4,' PLUS ',
WRITE(6,291) I,DLT
C
C PENALTY --- MAKE THE INTERVAL EVEN SMALLER.
DLT=DLT/STP
STP=STP*2.
ITRY=ITRY+1
55 CONTINUE
C
C
C 292 WRITE(6,292)
WRITE(6,292) I, A SOLN IS NOT POSSIBLE. THE PROGRAM SURRENDERS.'
C
C STOP
C
C 65 CONTINUE
I=I + DLT
ITRA=ITRY+NJMEQ*ITRY
WRITE(6,350) I,DLT,ITRA
350 FORMAT (5X,I=,G12.6,4X,'DT= ',G12.6,4X,
1 'ITERATIONS REQUIRED =',I3)
C
C IF THE SOLN REQUIRED LESS ITERATIONS THAN PREVIOUSLY,
INCREASE THE SIZE OF THE INITIAL TIME STEP.
LRGE = ITRA
IF (ITRA.LT.LRGE.OR.ITRA.LT.MIN) DLT=DLT*1.2
C
C IF ((I-PTIME)/PTIME .LT. STAR) GO TO 377
PTIME=I
WRITE(6,354)
354 FORMAT(/5(4X,'NODE',7X,'PSI',4X))
DO 355 I=1,NUF
I1=1+NUF
I2=I+2*NUF

```



```

I3=I*3*NUF
I4=I*4*NUF
WRITE(6,356) (I,PV(I),I1,PV(I1),I2,PV(I2),I3,PV(I3),I4,PV(I4))
CONTINUE
355 FORMAT(5(4X,I3,3X,IPE12.4))
356 WRITE(6,357)
357 FORMAT(/10X)
700 FORMAT(/1X,VALUES OF IPT-1, IPT-2, IPT-3,/)
C
NIT= I + NIT
IF (NIT.GT. 400) GO TO 377
377 CONTINUE
C
DO 27 MN=1,NUMSNP
PS1IV(MN)=PV(MN)
DO 28 I=1,LCON
BIGC(MN,I)=.0
28 CONTINUE
27 CONTINUE
IF (I.LT. IF) GO TO 351
778 FORMAT (2X,8(E14.5))
399 FORMAT (10X,I4)
WRITE (6,399) NIT
WRITE (3) BATCH
WRITE (4) BATCH
STOP
END

```



```

C-----DATA GENERATOR TO PROVIDE SYSTEM NODAL POINT
C          RADIAL AND AXIAL POSITION
C
      IMPLICIT REAL * 4 (A-H,O-Z),INTEGER * 4 (I-N)
      INTEGER * 4 SYSNOD,ELNOD
      DIMENSION R(15),Z(15),X(15),Y(15),SYSNOD(15),
1      ELNOD(220,3)
C      READ IN GRID VALUES AND NUMBER OF NODES
C
1      READ(5,100) NH,NV
100     FORMAT(2I5)
      WRITE(6,200)
200     FORMAT('1')
      WRITE(6,300) NH,NV
300     FORMAT(2X,'NH =',I5,5X,'NV =',I5)
      READ(5,400) (X(I),I=1,NH)
      READ(5,400) (Y(J),J=1,NV)
400     FORMAT(15F5.1)
      WRITE(6,500) (X(I),I=1,NH)
      WRITE(6,501) (Y(J),J=1,NV)
500     FORMAT(//2X,'X',5X,14F8.3)
501     FORMAT(//2X,'Z',5X,14F8.3)
C
      NVH=NV*NH
      NN=NV-1
      K=0
      DO 10 I=1,NH
      DO 20 J=1,NN
      K=K+1
      SYSNOD(K)=K
      R(K)=X(I)
      Z(K)=Y(J)
20      CONTINUE
10      CONTINUE
      K=NVH-NH
      DO 60 I=1,NH
      K=K+1
      SYSNOD(K)=K
      R(K)=X(I)
      Z(K)=Y(NV)
60      CONTINUE
      WRITE(6,700)
700     FORMAT(/////5X,'NODE',11X,'R',11X,'Z')
      WRITE(6,600) (I,R(I),Z(I),I=1,NVH)
      WRITE(7,601) (I,R(I),Z(I),I=1,NVH)
600     FORMAT(/5X,I5,2F15.5)
601     FORMAT(5X,I5,2F15.5)
C
      COMPUTE NODAL POINT CONNECTIVITY OF ELEMENTS
C
      LL=1
      K=1
      MM=0
      NHH=NH-1
      NVV=NV-2
      DO 30 I=1,NHH
      DO 40 J=1,NVV
      ELNOD(K,1)=LL
      ELNOD(K+1,1)=LL
      ELNOD(K,2)=MM+NV
      ELNOD(K+1,2)=LL+NV
      ELNOD(K,3)=ELNOD(K+1,2)
      ELNOD(K+1,3)=LL+1
      LL=LL+1
      K=K+2
      MM=MM+1
40      CONTINUE
      LL=LL+1
      MM=MM+1
30      CONTINUE

```



```

DO 70 I=1,NHH
  ELNOD(K,1)=NN
  ELNOD(K+1,1)=NN
  ELNOD(K,2)=NN+NV-1
  ELNOD(K+1,2)=LL+NV
  ELNOD(K,3)=ELNOD(K+1,2)
  ELNOD(K+1,3)=MM+NV
  K=K+2
  NN=NN+NV-1
  LL=LL+1
  MM=MM+1
  NEL=K-1
70  CONTINUE
  WRITE(6,750)
750  FORMAT('1')
  WRITE(6,800)NEL
800  FORMAT(5X,'NUMBER OF ELEMENTS =',I4)
  WRITE(6,950)
950  FORMAT(///10X,'CONNECTIVITY MATRIX')
  DO 50 I=1,NEL
    WRITE(6,900)I,(ELNOD(I,J),J=1,3)
    WRITE(7,901) I,(ELNOD(I,J),J=1,3)
50  CONTINUE
900  FORMAT(/4I10)
901  FORMAT(4I10)
    IF(NV.EQ.12)STOP
    GO TO 1
  END

```



```

C-----NUCLEAR PROPERTY DATA GENERATOR-----
C
      INTEGER * 4 NEL,ELEM,ITYPE
      DIMENSION V(220),ALPHA(220),SGF(220),SGA(220),D(220),
1      ITYPE(220),ELEM(220)
C      READ(5,100) NEL
      IDENTIFY CORE AND REFLECTOR ELEMENTS
100      READ(5,200) (ITYPE(I),I=1,NEL)
200      FORMAT(2I10)
      FORMAT(16I5)
      WRITE(6,400)
400      FORMAT('1',2X,'ELEM',6X,'D',8X,'SGA',9X,'SGF',8X,
1      'ALPHA',8X,'V')
300      FORMAT(2X,I4,5(1PE12.4))
C
C      ASSIGN PROPERTY VALUES TO ELEMENTS
C
      DO 30 I=1,NEL
      ELEM(I)=I
      V(I)=4.8E+07
      ALPHA(I)=1.0E-05
      IF(ITYPE(I).EQ.0)GO TO 10
      D(I)=1.2
      SGA(I)=.008
      SGF(I)=0
      GO TO 20
10      D(I)=0.913
      SGA(I)=0.014
      SGF(I)=0.008
20      WRITE(6,300)(I,D(I),SGA(I),SGF(I),ALPHA(I),V(I))
30      CONTINUE
      WRITE(7,500)(V(I),I=1,NEL)
      WRITE(7,500)(D(I),I=1,NEL)
      WRITE(7,500)(SGA(I),I=1,NEL)
      WRITE(7,500)(SGF(I),I=1,NEL)
      WRITE(7,600)(ALPHA(I),I=1,NEL)
500      FORMAT(8G10.5)
600      FORMAT(5F15.8)
      IF(NEL.EQ.64)GO TO 1
      IF(NEL.EQ.112)GO TO 1
      IF(NEL.EQ.220)STOP
      END

```



```

C -----DATA GENERATOR TO PROVIDE NODAL NEIGHBOR
C CONNECTIVITY
C
C      INTEGER * 4 NV,NVH,LCON,NN,MM,LL,JJ,MNOD,II,KK,
1      IJ,IK,IM,IN,NH,LI,LJ,LK
C      DIMENSION MNOD(132,7)
C
C      READ IN INITIAL DATA
1      READ(5,200) NV,NVH,LCON,NH
C      200      FORMAT(4I5)
C      COMPUTE MATRIX COUNTERS
C
C      M=NV+1
C      N=NV-1
C      NN=2*N
C      MM=3*N
C      II=4*N
C      KK=5*N
C      IJ=6*N
C      IK=7*N
C      IM=8*N
C      IN=9*N
C      LK=10*N
C      LI=NVH-NH
C      LJ=LI+1
C      LL=LI-NV+2
C
C      CALCULATE COMPACTED NODAL NEIGHBOR CONNECTIVITY
C      MATRIX
C
C      DO 10 I=1,NVH
C      MNOD(I,1)=I
C      MNOD(I,2)=I+1
C      MNOD(I,3)=N+I
C      MNOD(I,4)=NV+I
10      CONTINUE
C      DO 20 I=2,NVH
C      MNOD(I,5)=I-1
20      CONTINUE
C      DO 30 I=NV,NV+1
C      MNOD(I,6)=I-N
30      CONTINUE
C      DO 40 I=M,NVH
C      MNOD(I,7)=I-NV
40      CONTINUE
C      DO 50 I=1,NV
C      MNOD(I,7)=0
50      CONTINUE
C      DO 60 I=1,N
C      MNOD(I,6)=0
60      CONTINUE
C      MNOD(1,5)=0
C      MNOD(N,2)=LI+1
C      MNOD(N,4)=LI+2
C      MNOD(NV,5)=0
C      MNOD(NN,2)=LI+2
C      MNOD(NN,4)=LI+3
C      MNOD(NN+1,5)=0
C      MNOD(NV+1,7)=0
C      MNOD(MM,2)=LI+3
C      MNOD(MM,4)=LI+4
C      MNOD(MM+1,5)=0
C      MNOD(MM+1,7)=0
C      MNOD(LI+1,5)=0
C      MNOD(NVH,2)=0
C      MNOD(II,2)=LI+4
C      MNOD(II,4)=LI+5
C      MNOD(II+1,5)=0
C      MNOD(II+1,7)=0
C      IF(NVH.EQ.45) GO TO 15
C      MNOD(KK,2)=LI+5

```



```

MNOD(KK,4)=LI+6
MNOD(KK+1,5)=0
MNOD(KK+1,7)=0
MNOD(IJ,2)=LI+6
MNOD(IJ,4)=LI+7
MNOD(IJ+1,5)=0
MNOD(IJ+1,7)=0
MNOD(IK,2)=LI+7
MNOD(IK,4)=LI+8
MNOD(IK+1,5)=0
MNOD(IK+1,7)=0
IF(NVH.EQ.72)GO TO 15
MNOD(IM,2)=LI+8
MNOD(IM,4)=LI+9
MNOD(IM+1,5)=0
MNOD(IM+1,7)=0
MNOD(LK,2)=LI+10
MNOD(LK,4)=LI+11
MNOD(LK+1,5)=0
MNOD(LK+1,7)=0
MNOD(IN,2)=LI+9
MNOD(IN,4)=LI+10
MNOD(IN+1,5)=0
MNOD(IN+1,7)=0
15 DO 70 I=LJ,NV-1
   MNOD(I,6)=0
   MNOD(I,7)=0
70 CONTINUE
   DO 80 I=LL,LI
     MNOD(I,3)=0
80 CONTINUE
   DO 85 I=LL,LJ
     MNOD(I,4)=0
85 CONTINUE
   DO 95 I=LJ,NV-1
     MNOD(I,3)=N
     IF(I.EQ.NVH)GO TO 95
     MNOD(I+1,4)=MNOD(I,3)
     N=N+NV-1
95 CONTINUE
   MNOD(LI,2)=NV-1
   WRITE(6,300)
300 FORMAT(5X,'HEXAGONAL CONNECTIVITY',/' NODE')
   DO 90 I=1,NVH
     WRITE(6,100)(MNOD(I,J),J=1,LCON)
     WRITE(7,101)(MNOD(I,J),J=1,LCON)
90 CONTINUE
100 FORMAT(/9(2X,I3))
101 FORMAT(9(2X,I3))
   IF(NV.EQ.12)STOP
   GO TO 1
END

```


BIBLIOGRAPHY

1. Acton, F. S., Numerical Methods That Work, Harper and Row, 1970.
2. Arden, B. W. and Astill, K. N., Numerical Algorithms: Origins and Applications, p. 279, Addison-Wesley, 1970.
3. International Mathematical and Statistical Libraries, Inc. (IMSL), v. I, 1975.
4. Naval Postgraduate School Report 53Fe76051, A Program for Numerical Solution of Large Sparse Systems of Algebraic and Implicitly Defined Stiff Differential Equations, by R. Franke, May 1976.
5. Nguyen, D. H. and Salinas, D., "Finite Element Solutions of Space-Time Nonlinear Reactor Dynamics," Nuclear Science and Engineering, v. 60, No. 2, June 1976.
6. Olsen, R. A., Effective Methods for Solution of Nonlinear Reactor Dynamics Problems Using Finite Elements, M.S.M.E. Thesis, Naval Postgraduate School, Monterey, CA, 1975.
7. University of Illinois at Urbana Report UIUCDCS-R-73-575, Documentation for DFASUB - a program for the solution of simultaneous implicit differential and nonlinear equations, by R. L. Brown and C. W. Gear, July 1973.

INITIAL DISTRIBUTION LIST

	No. Copies
1. Defense Documentation Center Cameron Station Alexandria, VA 22314	2
2. Library, Code 0212 Naval Postgraduate School Monterey, CA 93940	2
3. Department Chairman, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	2
4. Associate Professor D. Salinas, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	2
5. Associate Professor D. H. Nguyen, Code 69 Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	1
6. Associate Professor R. Franke, Code 53 Department of Mathematics Naval Postgraduate School Monterey, CA 93940	1
7. LCDR R. C. Sheldrick, USN Naval Systems Engineering Division United States Naval Academy Annapolis, MD 21402	2

Thesis

168184

S44217 Sheldrick

c.1

A comparison of integration methods for the solution of non-linear reactor dynamics problems through the use of finite elements.

Thesis

168184

S44217 Sheldrick

c.1

A comparison of integration methods for the solution of non-linear reactor dynamics problems through the use of finite elements.

thesS44217

A comparison of integration methods for



3 2768 001 94417 6

DUDLEY KNOX LIBRARY